IN3007 – INDIVIDUAL PROJECT REPORT

Project Enigma – [redacted]
BSc (Hons) Computer Science 2019

Project Title: Natural Language Processing and Sentiment Analysis on Tweets in Real Time to Predict Influence of Tweets on Stock Prices

[redacted] [redacted]

Table of Contents

Chapter 1: Introduction

1.1 Project Overview

The project being described in this text (referred to as Enigma throughout the document) is an API which allows users to retrieve sentiment values and visualise these values over time regarding specified keywords as they are tweeted about on Twitter in real time. The intent behind Enigma is that it will allow users to evaluate these sentiments and be able to compare them to stock prices that relate to said keywords. The evaluation and analysis of Enigma is also carried in terms of accuracy and the data collection for this is in itself a component within Enigma.

1.2 Problem Description

The problem being addressed was whether, it is possible to process tweets that people make about a specific keyword in real time to determine if what they say in context of the keyword has an impact on the stock price of the company to which that keyword relates. In addition to this, if such a relationship exists, an evaluation of its nature was to be carried out. While Enigma is targeted at companies and products/services, it is still able to extract sentiment values on keywords that do not relate to service providers, e.g., “#Brexit” or “#BlackHole”. Enigma will enable interested parties to not only extract meaningfulness from data that has already existed but will also convert this data into actionable insights.

Enigma was originally intended to make recommendations and predictions regarding large stock price fluctuations (in the PDD), however, after completion and noting some observations (as will be seen in the Results section) this intent has now been changed to in addition to making the predictions, also understanding **how these fluctuations had occurred over time** as a result of how the sentiment on the specific stock had changed over time and **how these relate to each other**. In summary, the question being answered is “How can tweets be used to reliably predict the influence of people’s opinions (on a certain product/service) on the stock price of the product vendor/service provider, if at all it is possible to do so?”

1.3 Project Objectives

Main Objective: Enigma shall allow users to input search criteria (e.g., company/product names) into a tool and be able to make recommendations about the predicted imminent changes in stock price for that company based on Twitter tweets (if there are any) in real time.

The sub-objectives as specified in the PDD to achieve this were:

1) Enigma shall have a UI which will allow the user to input search parameters

Test: UI successfully accepts and transfers user input to backend

***Note**: This objective was dropped as the decision to make Enigma a consumable API instead of an application with a user interface was made. This allows for better flexibility in the was Enigma can be used and does not restrict the user to GUI functions made available.*

2) Enigma shall have a server to serve client requests as mentioned above

Test: Backend processes user input and sends back a response

3) Enigma shall carry out Natural Language Processing on tweets which match user's criteria

Test: Raw data is converted into a form which can be interpreted by sentiment analyser

4) Enigma shall extract overall sentiment (with confidence rating) about mentioned tweets and display this to the user visually on the UI

Test: Sentiment Analysis is accurate (can be tested by dummy data) and successfully returns a useful value

***Note**: While the UI was dropped, the graphically represented data is still available for observation.*

5) Enigma shall take advantage of containerisation (Docker) for running an isolated environment and to make maintenance and updating easy and clean

Test: The required Docker containers are spun up and run and each reflect the behaviour of the respective parts of Enigma that they will cover

6) Enigma shall exist live on the cloud, continuously monitoring certain keywords for sudden sentiment shifts, and alert the user if a large fluctuation occurs

Test: Emails are triggered when a fluctuation (over a pre-determined value) is observed on a particular keyword

Fortunately, all of Enigma's sub objectives (and so the main objective) have either been met as initially planned or changed to improve the efficiency and versatility of the way it can be used. How this was achieved did however change, and these changes will be found further in the report in the Methodology and Results sections.

1.4 Anticipated Beneficiaries

The details as to how these outputs can be used by the respective beneficiaries have been specified in the *Conclusions and Discussion*, section.
Individuals or organizations which will benefit from Enigma's completion:

- Further developers interested in Natural Language Processing and Sentiment Analysis which is carried out on vast sources of raw data such as Twitter in real time
- Researchers and individuals in statistics/finance who are interested in making relationships between tweet content, tweet frequency and stock price fluctuation within short time windows. Individuals in marketing may see the impact of an exceptionally good or bad demo about a product/service on Twitter
- Students and individuals of all ages interested in technology and finance who can understand what can be achieved by applying computer science to real world situations
- Artificial Intelligence and Machine Learning developers interested in using real values to make Enigma's prediction functionality more accurate by allowing it to "learn".

1.5 Work Conducted to Meet Objectives

In order for Enigma to reach the level of operation and functionality at which it is at, a series of components were carefully designed and implemented and then tied into each other and made to function seamlessly. These are all described in detail in the Results section. Each of these components functions independently with its own input(s) and is designed to output the desired data. The methodology chosen to do this was Agile, specifically Incremental (Scrum with a team) and Kanban. This will be described further and details as to why these were chosen, how they were adapted and how these were effective with respect to the nature and circumstances of the project in the *Methodology* section.

The components addressed one or more of the main objects as described above. The work involved having to research about and learn new technologies and design patterns and follow professional practices, none of which were taught at university. Every single

component integrated including programming languages and tech stack chosen were self-taught. Professional experience taken from placement year was also utilised in order to maintain professional documentation, coding and deployment standards.

Work conducted included using the Twitter API with tokens and keys on a python script which was programmed to get tweets and more importantly the tweet sentiments for a specific hashtag. In doing so a few interesting details were observe, and the work plan and objectives were accordingly modified to accommodate these for thoroughness. The tweets were processed to calculate their sentiments after a series of filtrations (special characters, profanity etc) and an algorithm was used to decide the weight each tweet was to have on the overall tag value. This involved making use of a classifier which will be described in detail further in the report under *Results*. A backend server was written from scratch in NodeJS (JavaScript) to handle requests made for specific hashtags and to allow the data retrieved from the python script to be inserted into a database (which is chosen to be SQLite for this purpose) and also to automate the python script to run on previously entered hashtags for values in the database to be updated every specified time period. In addition, the feature to send email alerts on large fluctuations periodically was also implemented. All of this was then brought together as a single application and bundled using Docker as a for Enigma to run as a containerised service. It was then taken to the cloud (AWS) to be deployed off of GitHub (where it was maintained) and run 24/7 to continuously monitor and update hashtag values in real time with added fault tolerance and availability. Finally, the use of a simple interface (provided by plotly) was employed to see the visualisation graphs regarding hashtag fluctuation history. All of these are explained in detail in the *Results* section.

1.6 Assumptions and Limitations

A major assumption made is that the relationship attempting to be discovered is approached correctly (i.e., the tweet processing in the way that it is done is the correct way). Another major assumption made, as with most data collection work, is that enough data was retrieved to make the analysis accurate. Finally, it is assumed that the analysis method used to compare the data and the way it was carried out was sufficient to reach a conclusion.

Enigma is only a prototype and solely exists for the purpose of demonstrating that it can be upscaled in all dimensions to extend functionality and utility. Enigma is heavily limited in computation power and the features chosen to be implemented. Since all APIs used were free versions, the number of API calls that can be made were limited versus a paid plan. Similarly, the cloud solution chosen was also a free one for the sake of demonstration and so

storage and processing power in this selection was also limited compared to a paid plan. Both these factors can easily be scaled up. Also, while Enigma focusses on only one dimension of real time data processing for predicting/understanding of user posts on stock prices, many more such sources can be extended onto the base application and several more types of data can be inferred from its use.

Chapter 2: Output Summary

Since Enigma is an application consisting of components, each of these components can be treated as a project output. The following are Enigma's outputs with brief descriptions. Details can be found in the results section to which they refer:

2.1 Python Script	
Description	The script responsible for retrieving sentiment values for the specified hashtags. **Input:** hashtag (String t) , number of tweets to get (integer x) **Output:** Array of x latest tweet sentiment values (floats) for tweets on t.
Type	Source code. 100 lines of code, all written by the developer. Makes calls to TextBlob and Twitter APIs (not in the source code).
Beneficiaries	Further Developers, Researchers, Students, Python Programmers. They may use this to study trends or to change how Enigma uses the script for gathering sentiment data
Link to Results Chapter	Chapter 5.2
Link to Appendix	Appendix B5

2.2 Node Application (Backend Server / API) and DockerImage file	
Description	The Node.JS application responsible for calling the python script above periodically and for carrying out CRUD operations on the database. Inputs and outputs (as well as automated actions) have all been documented as mentioned in the output to which 2.3 refers to below. The DockerImage allows the node app to be deployed with ease in a containerized environment.
Type	Source code. 567 lines of code in 4 files, all written by the developer. Utilizes the Express, AlphaVantage, SQLite, DateFormat, SendMail and Plotly node modules (not in the source code, these dependencies are installed and kept in a separate auto generated folder).
Beneficiaries	Further Developers, Researchers, Students, Node developers. They may use this to change the automated tasks which Enigma carries out and send various extended values to the database and python script from 2.1.
Link to Results Chapter	Chapter 5.3
Link to Appendix	Appendix B6

2.3 Node API Usage Documentation	
Description	A description of all the functions the node API serves with inputs and outputs and examples. Also describes the flow of actions that have been programmatically automated to allow Enigma to function.
Type	Swagger file. Utilizes the Express, AlphaVantage, SQLite, DateFormat, SendMail and Plotly node modules (not in the source code, these dependencies are installed and kept in a separate auto generated folder).
Beneficiaries	The beneficiaries for this output are the same as those specified in 2.2 above.

Link to Results Chapter	Chapter 5.5
Link to Appendix	Appendix B1

2.4 Enigma Component (Architecture) Diagram	
Description	Visual at a glance representation of the different components that make up Enigma and how they communicate and function with each other.
Type	Diagram, made using draw.io
Beneficiaries	The beneficiaries for this output are the same as those specified in 2.1 and 2.2 above.
Link to Results Chapter	Chapter 5.1.1
Link to Appendix	Appendix B3

2.5 Database Entity Relationship Diagram	
Description	Formal visual representation of the tables and columns that make up the SQLite database which Enigma utilizes and how they are related.
Type	Diagram, made using draw.io
Beneficiaries	The beneficiaries for this output are the same as those specified in 2.1 and 2.2 above.
Link to Results Chapter	Chapter 5.1.2
Link to Appendix	Appendix B2

2.6 Dataset	
Description	All the records generated by Enigma over a period of 2 weeks on miscellaneous hashtags, regarding how their sentiments have changed, most of which are technology companies. These values are time series and updated every minute.
Type	Dataset – time series, .csv
Beneficiaries	The beneficiaries for this output are the same as those specified in 2.1 and 2.2 above and in addition also includes data analysts or contributors looking to further analyse the data used to reach the conclusion in the Results section
Link to Results Chapter	Chapter 5.5
Link to Appendix	Appendix B4

Chapter 3: Literature Review

This chapter references various literature (books, papers and documentation) regarding topics relating to Enigma's development. This includes all stages including development methodology, design, software architecture, professional development practices and data analysis techniques. In addition, it includes some topics around the usage and existence of applications which are similar in nature to Enigma. Finally, it also includes references to various technology options which were either used in one of Enigma's components or were disqualified from being used due to unsuitability after evaluation.

3.1 Software Development Methodology

Initially, it was unsure which software development methodology would best suit the style of project that Enigma is and the single developer (team less, client less) nature of the development which was to be undertaken. There are several methodologies which were initially considered to work well for Enigma. These are all either a type of Agile methodology, or a very close relative of it. Initially, the short list included Scrum, Extreme Programming, Test Driven Development (TDD) and Kanban/LEAN. The evaluation for these and the approach selected for developing Enigma will be mentioned in the Methodology section, however the book "Agile Software Development" (Dingsøyr, Dybå and Moe, 2010) was useful in understanding how the word "Agile" is used as an umbrella term to cover different variants of methodologies which fall under it. Due to prior experience with Scrum (from placement year), the book "Software Engineering" (Sommerville, 2011) was also referred as it describes Extreme Programming, Incremental and Scrum and how to apply these to a project.

3.2 Natural Language Processing and Sentiment Analysis

Since Enigma operates over Twitter, a social media, the book "Sentiment Analysis in Social Media" (Pozzi et al., n.d.) allowed a better understanding of social media sentiment to be developed. The book covers topics such as sarcasm and opinionated statements (and as seen in results, this played an important role in the sentiment analysis function Enigma uses). The documents available at (Nlp.stanford.edu, 2019), especially the one on Tokenisation helped understand how statements can be tokenised and prepared to be fed into a classifier for processing.

3.3 Time Series Data Analysis

“Introduction to Time Series Analysis and Forecasting” (Yaffee and McGee, 2009) explains some common time series analysis concepts and how to carry them out using IBM SPSS. Enigma produces a data set of values generated through its sentiment analysis, and so these are analysed against a real data set for which this book was very useful.

“Introductory Time Series with R” (Metcalfe and Cowpertwait, 2009) was another booked referred to before the one mentioned above, but due to lack of understanding in R and the complexity of the analysis technique involved, this track was abandoned, and the IBM SPSS route was followed.

3.4 Technology

A variety of technological options were reviewed to make up the various components of Enigma. In the methodology section, it is explained why the ones which were picked were chosen due to convenience and intra application compatibility. However, for these decisions to be made, various official documentations along with their usage were reviewed. This included the documentation for the programming languages Ruby, NodeJS and PHP – as these are all web server compatible languages. For data gathering, Python and R documentations were reviewed, including the book on R mentioned in section 3.3 and Weka and IBM SPSS were evaluated as tools. HTTP response codes and what they mean and how they are used professionally were understood using official HTTP documentation. The decisions made to use the technology Enigma uses were made very carefully after a reasonable amount of evaluation and judgement. As much as possible, official and professional standards were maintained throughout Enigma’s technological implementation.

Once the appropriate decisions were made, the official language documentations for Python and NodeJS were referred to throughout the implementation process. In addition to these, “Python for Complete Beginners” (Jones, n.d.) and “Web Development with Node and Express” (Brown, 2014) were used for supplementary aid.

3.5 Similar Work in the Area

The paper “Trend Analysis of News Topics on Twitter” (Lu and Yang, 2012) proved to be a perfect starting point into Twitter fluctuations. The paper summarized how twitter reacts to news updates and how the trend originates and dissipates within Twitter. While this was in some way similar to the purpose Enigma is using tweets to predict stock movements, it did not prove to be valuable in understanding how to practically carry out the type of analysis required by Enigma. The paper focusses more on how and why trends occur and how to measure them and uses a concept called “trend momentum” to predict the “life” or dominance of a trend and also to predict an upcoming trend.

Another piece of work, “Sentiment Analysis on Twitter Data” (Bagheri and Islam, 2017) This paper especially helped gain an understanding on how to “clean” tweets i.e., remove what is not needed such as filler words, links etc. and work with what is leftover. Knowledge from this paper was used to inform a better tweet cleaning technique in Enigma’s python script. An interesting conclusion reached by this paper was that a significant proportion of the tweets analysed were labelled as neutral, which shows the limited nature of these classifiers and sentiment analysis techniques.

Both of these however, lacked the stock price context and had end goals different to those of Enigma. The first paper targeted news trends, and the second one was purely to run sentiment analysis following a methodical approach on tweets.

Chapter 4: Method

Chapter 4 outlines with details the approach taken to deliver each output and any changes made throughout the development period of Enigma. Each section of the methodology applies to one of the outputs as specified in Chapter 2. The sections do not describe what was done to make that output complete, but how it was done and what ideology was followed. The former will be found in Chapter 5.

4.1 Software Development Methodology

A brief discussion of the options considered to be used as methodologies for developing Enigma was made in Chapter 3 under Software Development Methodology. This subsection will explain the thought process behind selecting an appropriate methodology and how it was tailored to fit Enigma's development nature. It does not discuss individually what each of these are and how they all operate as this discussion falls outside the scope of this explanation.

The shortlist mentioned in 3.1 included Agile, Scrum, Incremental, Test Driven Development (TDD), Kanban/LEAN and Extreme Programming. It is important to note that Scrum, Incremental and Kanban are generally applied to a team of individuals and are not intentionally designed in context for solo developers. Scrum introduces the concept of sprints (time periods to achieve a sub goal) and daily scrum meetings. Kanban introduces the concept of having a board towards which contributors can collaborate and interact with. And Incremental is designed to take the waterfall approach of planning and break it down into do-as-you-go chunks.

For developing Enigma, Test Driven Development was primarily adopted, as this seemed to be the best fit for a 1-person project with known functionality and requirements (these will be discussed more in the section below) and with TDD knowing when a functionality is complete is made simple, since for Enigma there are no clients or stakeholders for feedback. If the test written for a functionality to be implemented passes, the functionality can safely be assumed to be marked complete. Enigma, however, also made use of various attributes of some of the other methodologies mentioned above. Even though these are all designed to be used in a team, they were taken into context to be used by 1 individual. For example, the Kanban board was still used (with the aid of the tool 'ZenHub' on within GitHub – where the software was also maintained) to monitor which features had been done, which features were awaiting testing, logging bugs encountered, marking tasks as complete and being aware of which tasks were currently being worked on. The idea of sprints was taken from Scrum, to help

attach deadlines to each task within a sprint and to reach an outcome at the end of each sprint. The work plan section below breaks down how this was initially designed. Each task was attached an estimated “point” value. The points were how many hours of work each task required and so these were used to make up the total time in a 2-week sprint and allocated per sprint accordingly. The period was chosen to be 2 weeks because, given the granularity to which tasks were broken down, which is considered good practice (because less can go wrong with overrunning a smaller time estimate than over running a large one) very large sprints were not required. Since each issue was based on a requirement (deliverable) and since each deliverable was different in terms of importance, complexity and past experience, the time factor for each was significantly different and so this method aided in the success of task completion. These estimates were not intended to be 100% accurate, however there was always room to accommodate an expected delay if there was one for specific features which required researching topics that were new. The accuracy of prediction regarding time points and mapping them onto sprint duration allowed a further improvement in judgement when deciding point values for new tasks every sprint, which was a crucial feature this approach enabled. Each task on the board was a component from the work plan (from the PDD, mentioned in the section below) and subsequent tasks were made to further break the component down technically if needed. With all of these brought together on the Kanban board, an at-a-glance dashboard view of all the tasks being worked on, not yet begun, completed and awaiting testing along with time estimated was available to be seen to monitor development and to increase time awareness. Finally, the idea of quickly developing very small components and testing them before moving forward was taken from Extreme Programming. In summary, Enigma does not use 1 specific methodology, but instead uses various attributes from suitable methodologies which fit the nature of Enigma as a project and combines them into 1 bespoke approach.

4.2 Work Plan

4.2.1 Work Plan Creation and Outline

Using the approach mentioned in 4.1, a work plan was constructed highlighting all the key components which were necessary for Enigma to be delivered as described. There were also some additional components added for after this stage which were aimed at exploiting computing procedures to further enhance the range of functionality and ability which Enigma could provide. There were a number of reasons due to which some of these could not be implemented, and instead of these some other ones were added instead. These will all be discussed in the Changes Made section below and the Results section.

Below is a table outlining what these initial components were, and they were given sprint allocations and time estimates using the component-point estimation technique as mentioned above in 4.1:

Sprint	Deliverable	Date
1	Backend server-side logic	15 Oct 18 – 31 Oct 18
2	Front end User Interface	1 Nov 18 – 15 Nov 18
3	Explore Twitter API, Research NLP procedures	16 Nov 18 – 30 Nov 18
4	Implement basic NLP and have it working on tweets	1 Dec 18 – 15 Dec 18
5	Research SA procedures, begin implementation	16 Dec 18 – 31 Dec 18
6	Continue SA implementation, test and improve against dummy data	1 Jan 18 – 15 Jan 18
7	Containerize application, move to cloud	16 Jan 18 – 31 Jan 18
ENIGMA BASIC REQUIREMENTS MET		
8	Explore AI, machine learning, neural networks	1 Feb 18 – 15 Feb 18
9	Begin integrating findings from sprint 8 to improve SA, check against dummy data and to train the model	16 Feb 18 – 28 Feb 18
10	Use findings from sprint 8 to enable prediction feature and repeat items from sprint 9 on prediction	1 Mar 18 – 15 Mar 18
11	Implement feature to alert user on large fluctuations by continuously checking live tweets	16 Mar 18 – 31 Mar 18

How these were approached and individually enforced are discussed in the following subsection.

4.2.2 Work Plan Application and Changes Made

The list below further elaborates the approach used and work conducted each sprint including any changes made and reasons behind why these changes were made:

Sprint 1, 2 (mid Oct – mid Nov): Understanding the technologies being used, for the server/backend and bring them together as a working web app. Installing all necessary software, packages and plugins etc. The UI was decided to be pushed back further as database setup and database logic on the server side needed to be done with a higher priority as this was missed out in the original plan due unawareness.

Sprint 3, 4 (mid Nov – mid Dec): Explored Twitter API, Explored and implemented Natural Language Processing techniques and procedures using a Perceptron (Neural Network Classifier) after statement tokenisation. Retrieved tweets on a keyword (hash tag) and had basic NLP running on them. This pulled back a feature from sprint 8 to earlier in the project.

Sprint 5, 6, 7 (mid Dec – end of Jan): Sentiment analysis (SA) attempted on output from sprint 4 above. Perceptron classifier tested with sentiment analysis, and evaluation resulted in very inaccurate classification outcomes. The current classification method had to be discarded and replaced with a new more accurate and reliable technique. This also pushed back the deployment to cloud and containerisation tasks further back, however this was acceptable as these 2 tasks were only most appropriately implementable in the final stages of Enigma's completion regardless.

Sprint 8, 9, 10 (Feb – mid March): Completion of sprint 7 yielded a new way to carry out sentiment analysis. This used a Naïve-Bayes classifier, the operation of which was outsourced to the Python TextBlob API. Because of this change in the classifier and the output type that was now being returned, changes needed to be made on the Node Application and database structure along with the database logic. Created containerisation and prepared cloud environment for deployment.

Sprint 11 (mid-March – end of March) and first half of April: (this sprint was originally kept as contingency and proved useful due to the project changes mentioned above). Finalised Enigma's basic functionality. Implemented the additional feature of alerting the user on live large fluctuations on predefined keywords. Added feature to visualise fluctuations on a graph. The AI aspect could not be implemented due to a blockage by how stock labels for a company were to be retrieved. This issue (and how it was addressed) is discussed in detail in Results. Manual intervention by the user to add the label was the workaround which was implemented. Conducted data analysis of findings.

4.3 Requirements Documentation and Completeness Tracking

As mentioned in 4.1, since there were no clients or direct stakeholders involved in Enigma's development, all requirements were self-presented. The requirements were represented as features and added to the Kanban board. Each requirement (feature) before implementation had a predefined test to pass. Development would then commence with the objective being to make the test pass. Additional testing was put in place, this is described in the Testing section below. Once all tests had passed, the feature was marked as complete. There were a few occasions where the test would not pass, or a better alternative was discovered. In this case, the necessary changes were made on the item description in the Kanban board. This

way, there was a history of changes and events maintained for every feature and this aided in the implementation of other components.

4.4 Architecture / Design

The need to create Design-Class diagrams and Use-case diagrams could not be justified for Enigma's design stage and so, was disregarded. This was due to the following reasons:

- Enigma did not follow an Object-Oriented approach technologically and so the concept of classes was non-existent. This disqualified the use of a Design-Class diagram for visualising the architecture.
- There were no real stakeholders for the project there was no need to formally document use cases and features via a use case diagram. The Kanban feature documentation approach seemed suitable on this occasion.

Since Enigma is aimed to be used as an API and does have users, all of whom can do the same things in terms of interacting with Enigma, an API specification document which outlines what services Enigma can provide as an API and how requests (what format) can be made to it. This can be found in Appendix and is described in the Results section.

To visualise the architecture of Enigma's operation, a component diagram was created to help users and interested parties understand how the components interact with each other and how Enigma as a whole functions. This too, can be found in Appendix and is described in the Results section.

In addition, an entity-relationship diagram was constructed to visualise how Enigma's (fairly simplistic) time series database is structured and how the various tables link with each other. This will be found in Appendix and further explained in the results section.

4.5 Implementation and Maintenance

As mentioned in the literature review section, for the technical implementation of Enigma, various types of technologies and programming languages were observed. Through proper judgement and the ability to map what is needed to be done to how it should be done, the correct decisions were made as to what technology/programming language was made responsible for which component within Enigma.

4.5.1 Development

This section outlines the decisions made regarding the different components with a development point of view. The individual outputs which these helped develop are described in detail in the Results section.

4.5.1.1 Server / API

Due to its asynchronous single threaded nature, NodeJS was chosen as the backend for the application. The server is responsible from taking requests from the user and returning the appropriate information. It was designed with proper asynchronous programming practices and also automates a series of tasks which Enigma needs doing. Node is suitable for this as well due its openness to event driven programming. The Node application is also what queries and makes transactions to the database. Lastly, Node was chosen for its ability to spawn child processes and this was useful to run the python script and use its output within the application. Features implemented in the Node application are detailed in the results section.

4.5.1.2 Sentiment Analysis

Initially, both R and Python were being evaluated for their ability to repeatedly carry out a series of operations very effectively especially in the context of data processing. Due to simplicity of the language and ease of access to several APIs (such as TextBlob) Python was chosen. This also proved to be a useful and enjoyable learning experience. Python documentation and the book mentioned were resourceful to maintain convention and python best practices.

4.5.1.3 Database

For the database (which was to be time series) SQLite was chosen for its lightweight and simplistic nature. It does not require complicated connections to be made and maintained. A single threaded web request interaction style such as the one used by Node (alongside the SQLite module) allows data flow to and from the database to be made seamlessly without concerns for concurrency, as this is handled outside the database itself. The choice of SQLite also allows full flexibility in terms of the database being used in different environments, for e.g., locally, in a container or on the cloud.

4.5.1.4 Modules, APIs and Libraries Utilized

Several modules were made use of from the Node Packaged Modules (NPM) repository – these are available online and free to use for educational and non-profit purposes as part of an open source agreement. The reasoning behind this is that there is no need to recreate technological components which have already been created and mastered and are continuously maintained by a community. When these are used, there is much better chance that their development process will be much smoother and less likely to create blockages, and in addition the quality of the software program improves. These are Plotly to generate visual graphs, Express to handle the web request routing, SendMail to send emails to users, DateFormat to format date objects, AlphaVantage (a module to call the AlphaVantage stock API). These will all be referenced at the end.

For Python, the twitter API and TextBlob API were used.

In addition to these, various libraries from the Node and Python standard libraries (included with the language) were also utilized. The specific imports regarding all of these can be seen on the top of both source code files.

4.5.2 Maintenance

All source code (and auto generated code) was maintained on GitHub, which is a well-known version control system. GitHub is free to use for students and allows the creation of private repositories, one of which Enigma's source code was maintained in. In addition, GitHub allows the ZenHub plugin to be installed in browser and allows the creation of the Kanban board mentioned in 4.1 within GitHub. This also allows seamless linking of the Kanban board tasks with features within the code. For example, a specific code push (commit) made to GitHub had the ability to close the related task on the Kanban board automatically.

4.5.3 Deployment

On completion, and to run the experiments mentioned in the results section and below in 4.7, Enigma was containerised using Docker and deployed to AWS (Amazon Web Services) free cloud where it was left to run and collect data. Containerisation is considered professional deployment practice for service disposability and ease of independently being run in any environment from a DevOps (developer operations) point of view.

4.6 Testing and Evaluation

As mentioned above, since the bespoke methodology followed for developing Enigma inherited heavily from Test Driven Development, testing was a crucial part of making sure each component was successfully and wholly complete. For all features, a test was written

which would expect an outcome specific to the feature, and coding was then initiated in the opposite direction to create a function to make the required output and so make the test pass. The feature or component was only considered complete when the prewritten test passed and also when any invalid or extreme inputs were also treated as expected. This was a sort of regression testing. If any tests did not pass, the function was modified, and the test retried iteratively until they did. For any bugs discovered post component completion, these were logged and a short description of why and how (for e.g., which input can recreate the bug) it occurred. This knowledge was used as a learning mechanism to further improve future pre-implementation tests. For unfamiliar topics, research was done on common testing procedures and practices and these were included as part of the tests.

This testing methodology helped identify the classifier issue and stock automation issue as mentioned in the Work Plan Changes.

4.7 Experimentation and Data Analysis

A large part of the results section addresses the analysis of the data and the experiment conducted to do this. The methodology followed here was that once Enigma was completed and ready to be deployed, it was set up on the cloud and given 7 tags (limited by the number of calls able to be made by the free Twitter API). These tags were IBM, Microsoft, Apple, Walmart, Google, Tesla and PayPal. For each of these tags, the per minute sentiment fluctuation was stored into the database. In parallel to this, the stock market API (AlphaVantage) was also called every minute for the company equivalent each of these 7 tags and the stock prices recorded. Since the stock market is only open from 9:30 AM to 4:00 PM on weekdays Eastern Time, this did limit how much of Enigma's data can actually be compared with real stock data, however, given the number of points involved (1 per minute for 6.5 hours is 390 data points per tag per day) this should not be a huge problem. Of course, as is the case with any data analysis, in an ideal world, the model could be run for months and could be allowed to undergo various seasonal and event-based changes. However, in the timescale of the project this could not be made possible. More limitations and assumptions will be discussed in the conclusion. Once the 2 data sets (enigma values, and stock price values) were gathered, they were fed into IBM SPSS where a time series data evaluation was carried out. The conduct and results of the experiment will be discussed in the Results section.

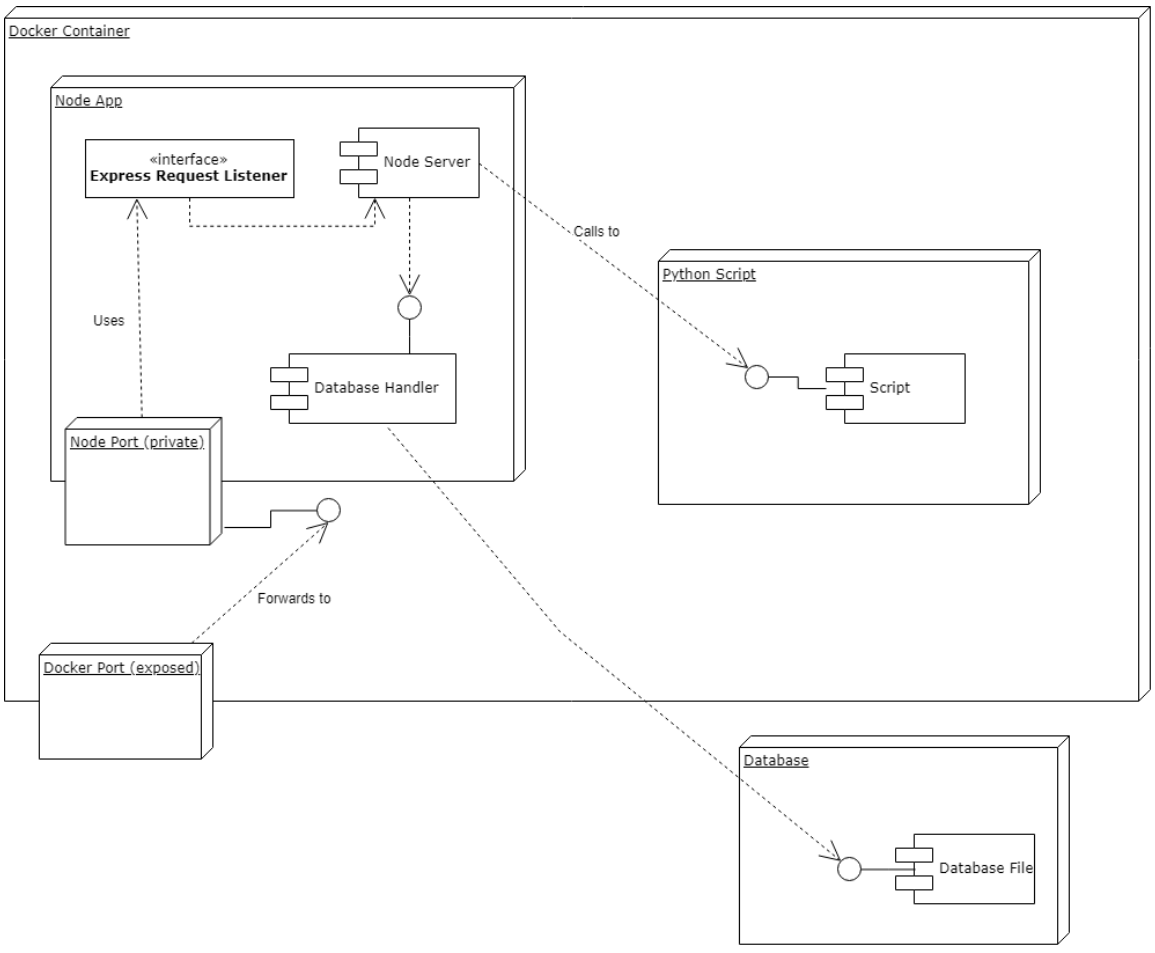
Chapter 5: Results

This chapter details all outputs described in chapter 2 the output summary and explains how these were made and what value they add to enigma's completion. It will outline and explain all technological decisions made on a feature and the work done to make it complete. Towards the end, this chapter also addresses the experiment mentioned in the end of chapter 4 and breaks down the analysis and the outcomes observed.

5.1 Design and Architecture Diagrams

5.1.1 Component Diagram

After having decided which technology/programming language to use to build up the various components it was crucial to determine on a top level how these components would communicate with each other. Below is a component diagram which displays how Enigma was decided to be built. Although a brief explanation of how the system works as a whole will be provided in this section, details for each specific component in the diagram and what makes it up and how these carry out the required tasks will be provided in their respective sections in this chapter.



Referring to the diagram, it can be seen there are 2 external components on the highest level which make up Enigma - the docker container and the database. The docker container further contains 2 sub components – the node app and the python script. Since the components inside the container are in their own network and linked in a way which makes them unreachable from outside the container, the docker container exposes a port which allows the user to send requests to it via a browser or using an HTTP request tool such as Postman or CURL. This port then maps onto a port on which the node app listens and so user requests get forwarded to this port for something useful to happen to the users request, i.e., the node app accepts it. The docker file which specifies how the containerisation happens and how the order of events take place is described in 5.4. The format in which request must be made to Enigma follow standard HTTP convention and are further specified in the API Documentation Appendix.

Inside the container the Node App is responsible to take the request and, in most cases, do 1 of 2 things based on the request:

- Either the user is requesting data from the database and/or asking it to be visualised

OR

- They are entering a new tag onto the Enigma database, in which case the tag is passed on to the Python script

In the first case, if they have requested data or a visualisation (graph) on fluctuations for a tag, the node server accepts this request and retrieves values from the database via the database handler and sends it to the plotting library for processing (detailed in section 5.3).

If the user is entering a new tag to be added for monitoring into Enigma, the server requests the tag sentiment values from the python script (detailed in section 5.2) and stores them in the database via the handler.

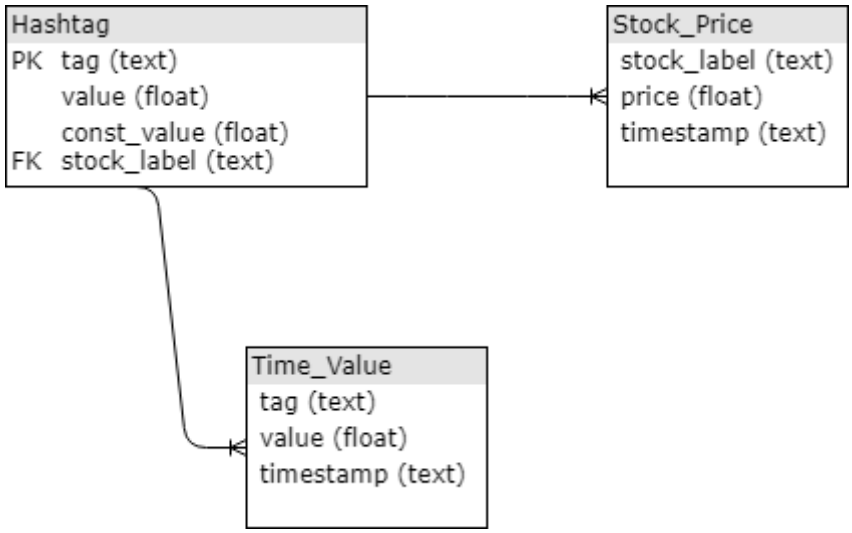
The way Enigma is designed, the only way the database file can be accessed is through the database handler, and this is only reachable once a request has been evaluated to be a valid and in the correct format. The python script can also make changes to the database, but this is done again through the Node app via the database handler. This is considered good professional practice as it limits the ways a resource can be accessed or used.

Finally, a major architectural decision made (which may have been observed in the diagram and stood out) is that the database resides and interacts with the container components from outside. The reason this decision was made was because the whole point of containerising applications is to make them more disposable and transportable. For example, every time a

change is made in the application, the old container can just be replaced with a new one containing the updated application. If the database was also inside the container, all previous data would be lost along with the container when it gets disposed. The idea of keeping the data store outside the container is called “persistent storage” as the data store persists and stays the same even if the container has been disposed.

The entire setup was then moved onto AWS Free Tier cloud so that it can be run 24/7 without interruption, as on a local machine it would need to stay on continuously. The computational power on AWS was also better and slightly quicker, although this was not a huge problem.

5.1.2 Entity Relationship Diagram



The diagram above describes the relationship of tables and columns with each other in the simple time series database which Enigma uses. The main table is the Hashtag table, this stores the actual tag itself, which is a unique primary key for the Hashtag table and also a key used in the Time_Value table. These two tables have a 1-many relationship, i.e., 1 tag in the Hashtag table has multiple time series records in the Time_Value table. The value is just the number returned by the python sentiment analysis script for that tag. The const_value field is also a number, but this is the value for that tag over a constant period of time. The need for this and its operation will be further explained in 5.3. Finally, there is a stock_label field, and this is the stock market label for the company which the tag refers to. For example, if the tag being searched on twitter is “Google”, its stock market label is “GOOG”. These labels are all predefined in the stock market and are specified by the user. More details on this will follow in 5.3. Once this connection has been made, there is another 1-many relationship between the Hashtag and Stock_Price tables, i.e., 1 stock_label in Hashtag has multiple stock price records in Stock_Price. Finally, the Stock_Price table just holds time

series data on the value of the stock prices for all the tags with specified labels in the Hashtag table.

The SQLite Database Design Language (DDL) is run at the start of the node app to create the tables:

```
function initialiseDatabase() {
  db.serialize(() => {
    db.run('CREATE TABLE IF NOT EXISTS Hashtag (tag TEXT UNIQUE PRIMARY KEY NOT NULL, value INT NOT NULL' +
           ',constant_value INT NOT NULL, stock_label TEXT);');
    db.run('CREATE TABLE IF NOT EXISTS Time_Value (tag TEXT NOT NULL, value INT NOT NULL, timestamp text NOT NULL);');
    db.run('CREATE TABLE IF NOT EXISTS Stock_Price (stock_label TEXT NOT NULL, price INT NOT NULL, timestamp text NOT NULL);');
  });
}
```

5.2 Python Script

5.2.1 Overview

The first output as mentioned in Chapter 2, 2.1, is the python script which conducts the sentiment analysis. As described before, the inputs to this script are:

- A tag, in string format, for which the sentiment values are to be returned from twitter
- A number, integer, which specifies how many of the recent tweets using the tag to pick up for analysis from twitter. When the Node app calls the script, it uses a set value of 10 tweets.

Note that the number of tweet searches which can be made using the free Twitter API in a given time period is fixed and making any additional requests which exceed this number within the time window are simply rejected.

The output of this script (which is used by the Node app) is:

- An array containing the values of the sentiments for the last 10 tweets (or n tweets if the python script is run directly, where n is specified by the user)

Now that the inputs and output have been explained, the processing which takes place in between is the most significant part of this component.

Firstly, authorisation is setup with the Twitter API for making the search requests using the “tweepy” library. The OAuthHandler object is responsible for taking the access tokens and authorisation keys provided by twitter for the account and for authenticating the request.

5.2.2 Tweet Cleaning

After this is ready to be used, the script then passes on the tag and the number of tweets along with the authentication object to the API and if the authorization is successful, an array

of the requested tweets is returned. The API is set to make sure only English tweets are returned and profanity is removed.

The tweets are then cleaned based on the following regular expression:

```
def clean_tweet(self, tweet):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^#0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
```

This regular expression removes any special characters and links which may interfere with the sentiment analysis. The sentiment analysis is most accurate with plain English and clean sentences and this is why these steps are taken.

5.2.3 Sentiment Analysis

At this point, it seems appropriate to mention there was a different approach taken to the one currently used by Enigma regarding Natural Language Processing (NLP) and Sentiment Analysis initially. The approach involved tokenising (splitting the sentences into processable chunks, words), and then running them through a Perceptron (a neural network-based classification model). The tokenisation and neural network were both provided by python libraries. However, it was quickly discovered that this approach did not yield accurate natural language processing results and so the sentiment analysis during the evaluation process, by human judgement, seemed to be obviously and extremely incorrect. Because of this, the classification and sentiment analysis approach were revised, and a better solution was used which proved to be much more accurate (although still far from perfect). This is the current approach and will now be described.

To replace the inaccurate method, the sentiment analysis was outsourced to the TextBlob library. This yielded much more accurate results as compared to the Perceptron based method, again evaluated by human judgement, and was set as the method going forward.

Initially, just the sentiment polarity, which is how shifted towards negative or positive a particular statement or set of statements is, was used to determine the overall sentiment of the tweet. While this did work, it was later discovered that statements had another property called subjectivity, which described how subjective they were. Polarity is a number from -1 to +1, with a more positive value being a more positive polarity. Subjectivity is a number from 0 to 1, with 1 being a very subjective statement. An example of a positive polarity statement is “I think they did an absolutely fantastic job. Well done team!”. This statement holds a value of +0.9, meaning it is 90% positive and has a subjectivity of 0.9, so it is also very subjective. A statement like “The sky is blue” has a polarity of 0, which means it is neutral and a subjectivity of 0.3 which means it is not very subjective. Although these examples are accurate and provide a better understand of how Enigma’s sentiment analysis function

(described below) works, it is important to understand that in the real world (and on social media) sentences are rarely this short, simplistic and clear in terms of sentiment evaluation, especially by a computer, and so this level of accuracy is hardly maintained.

```
def get_tweet_sentiment(self, tweet):
    analysis = TextBlob(self.clean_tweet(tweet))
    subjectivity = analysis.sentiment.subjectivity
    polarity = analysis.sentiment.polarity
    sentiment_value = polarity * subjectivity
    return sentiment_value
```

Shown above is the function which gets the sentiment for each tweet based on the understanding described above. The value for sentiment is the product of the tweet's polarity and subjectivity. In a way, the polarity of the tweet is given less or more weight based on how subjective it is. The reasoning behind this is that, more objective tweets tend to state facts and so when facts are being stated (although wrong, the risk is taken) they tend to be more accurate as opposed to opinions or personal experiences.

The value is then stored in the array, the process repeated for the other tweets, and then the array returned, ready to be used by the Node application for further processing.

5.3 Node Application

5.3.1 Outline

This is the component responsible for doing most of the work which Enigma carries out. The following is a description of all the endpoints Enigma accepts requests on as an API, what format the requests must be in and what these endpoints return:

/tag/add along with a query parameter which is the tag needed to be added onto Enigma. The result is that the tag is run through the Python script and the value stored in the database so that it can be updated every specified time period.

/tag/view along with a query parameter which is the tag that is already on the database, which has been maintained. This plots a graph to show the user with all the value points stored by enigma on that tag over time. A link is provided to the user to see the graph.

/stock/add along with a merge parameter in the format <TAG>*<STOCK_LABEL>, i.e., two strings with asterisk in between. This allows the user to link a tag in the database with a stock label (found by the user) so that the stock prices for the tag can also be monitored in the same time period for comparing against enigma's values.

/stock/view along with a query parameter which is the stock label which is on the database, whose price has been followed. This plots a graph to show the user with all the price points stored by enigma on that stock label over time. A link is provided to the user to see the graph.

In addition to these, a large part of the Node application's functionality is automated and continuously running in the background. This will be explained in 5.3.3 below.

5.3.2 Features

A note to be made regarding all features within the Node application is that it uses asynchronous programming functions. This means it operates on a single thread (except for when it calls the python script) to do everything. This means that it incorporates the use of callback functions. So, put simply, when an asynchronous function is called, it is left "unaccounted" for, and when the function has completed its task(s), it "calls back" to the function which called it with the output. This is extremely useful for tasks which require time such as disk I/O, database transactions or API data requests etc. The use of these functions was made as much as possible to make sure that the application runs with optimization and also follows professional node programming practice.

A series of checks and evaluations are also made on inputs and the validity of the inputs to make sure that

1) nothing the user can do will cause the server to crash in terms of the input they provide

and

2) the database is only contacted once preliminary checks are made and this operation has been justified

This is for eliminating problems within the application for a smoother operation and also so that no errors are returned from the database and are instead returned from the application. One such check which uses callbacks (and is used generically almost every time the user interacts with the node application is the checkIfTagExists() function:

```
function checkIfTagExists(tag, callback) {
  db.all(`SELECT tag FROM Hashtag WHERE tag = ? COLLATE NOCASE;`, tag, (err, response) => {
    if (err) {
      return callback(err, false);
    } else if (response && response[0] !== undefined) {
      return callback(null, true);
  }
    return callback(null, false);
  });
}
```

5.3.2.1 Adding Tags and Stock Labels

As mentioned in the overview, calling the correct endpoints and using the right format allows these 2 functions to be carried out. Both of these features make use of the various validation techniques and callback conventions mentioned above. Adding the tag runs the script on it and puts in the database. Adding a stock label and merging it with a tag creates the link so that data about stock prices for that tag can also begin to be collected. An example of both of these being done on a tag is shown below:

	tag	value	constant_value	stock_label
	Filter	Filter	Filter	Filter
1	apple	0.05525	0.0455578512...	AAPL

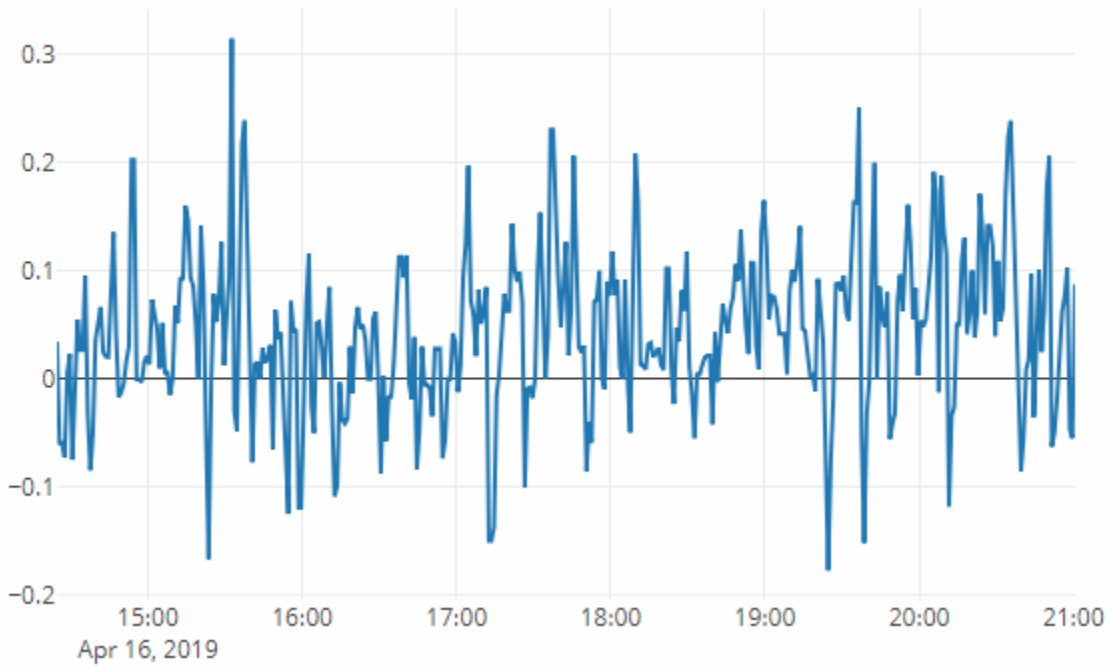
(If a stock label has not yet been merged with the tag, this value remains null until this has been done)

The database state visualisation was made using a tool called "DB Browser for SQLite".

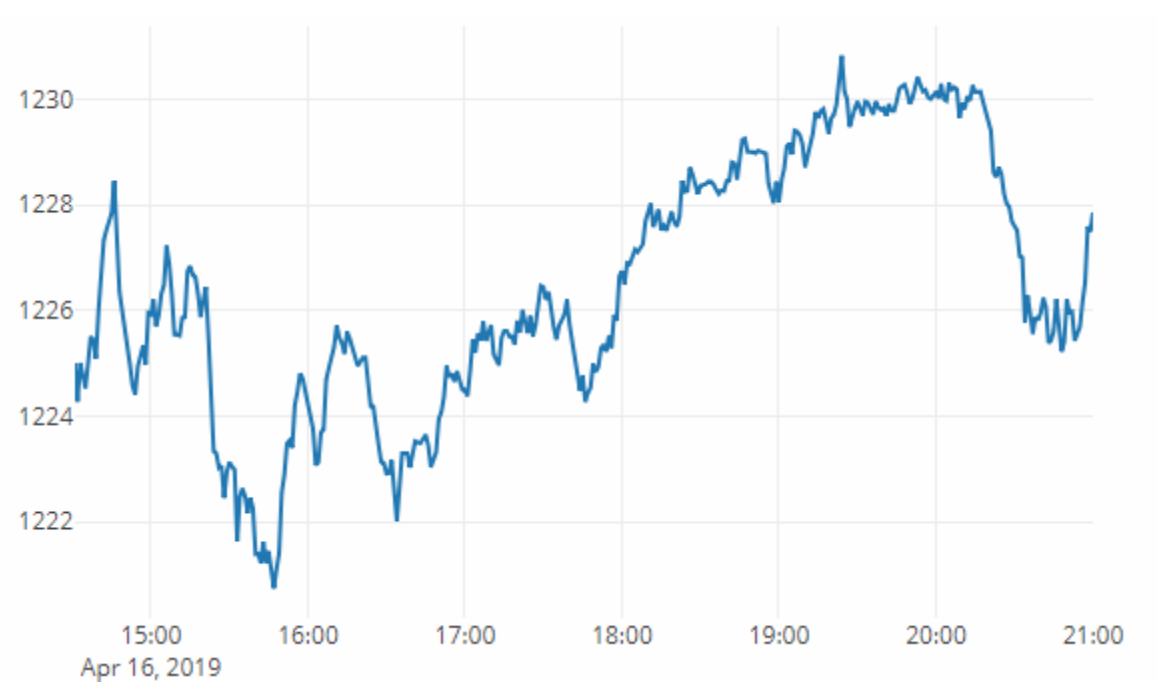
5.3.2.2 Viewing Graphs for Stock Price History or Tag Value History

The node module used to create the graphs is called "Plotly". Once the appropriate request is made, all values are retrieved from the database for the specified keyword and put into 2 arrays, the timestamps go onto the x-axis and the values on the y-axis. These are then handed over to the plotly module for processing. The output for each of these can be seen as examples:

Enigma tag value (for tweet sentiments) for the word "Google" –



Stock price fluctuation graph made (using the AlphaVantage API for prices) for the stock label “GOOG” (stock label for Google) for the same times as shown above–



5.3.3 Automated Actions

Although these aren’t user triggerable features, these actions run in the background and are very crucial to the functioning of the node app. These involve a series of fairly complicated operations all repeated over a set time of 1 minute. This frequency was perfect as it did not

clash with the maximum API calls allowed for various APIs within the time window, as free versions for all APIs were used. It also made sure that all computational processes could be completed well in time for the next cycle on a computationally limited device.

5.3.3.1 Tag Sentiment Value Updating and Adding of Records

The first task addressed in the automated actions function was to retrieve all the tags from the database and them into an array. Next, a child process was spawned (to execute an action in parallel) for calling the python script with each of the hashtags in the array and the value 10 to retrieve 10 tweets per tag and so the 10 sentiment values per tag.

```
function callScript(query, res) {
  console.log("Script called");
  query = `#${query}`;
  const pythonProcess = spawn('python',["./twitter.py", query, 10]);
  pythonProcess.stdout.on('data', (data) => {
    res.send(`Here are tweet sentiments for ${query}:${data}`);
  });
}
```

Next, the values for each tag are averaged (by summing the values and dividing by the number of elements in the array) and an update function is called for each tag to do 2 things:

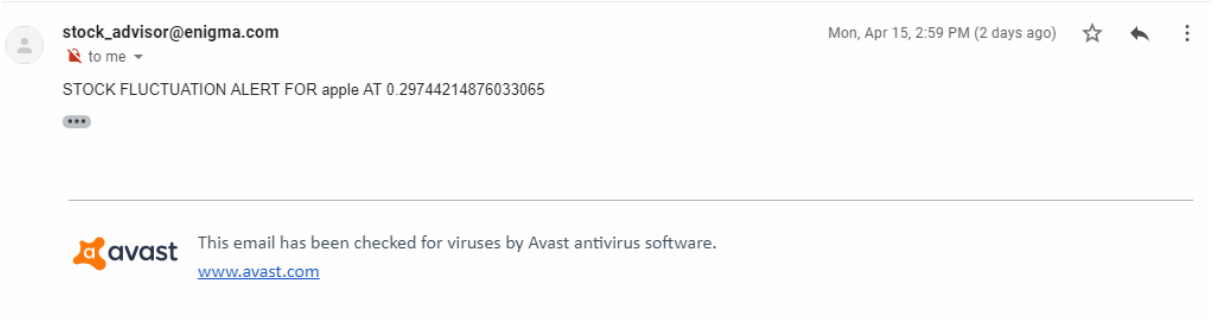
1) Update the value of the tag in the Hashtag table to reflect current value
2) Add a record with a timestamp of the value in the Time_Value table to make historical entries (for creating the over-time graph)

The date objects are also formatted and stored in the “yyyy-mm-dd HH:MM:ss “format for professional practice, compatibility with the plotly module and for consistency.

5.3.3.2 Constant Value Monitoring and Alerting

Another automated task which makes Enigma useful is the sending of an email when a fluctuation within a specified time (which was set to be 2 hours) is observed. When the values for each tag are retrieved as mentioned above every minute, the value being updated is compared with the const_value, and if there is a fluctuation (set to be more than 0.25) the email is triggered via the SendMail node module.

The email contains the name of the tag along with the fluctuation magnitude:



The function which evaluates the difference in tag values and sends an email if the threshold has been reached:

```
function evaluateTagDifference(tag) {
  return database.calculateTagDifference(tag, (err, difference) => {
    //difference = tag.constValue - tag.currentValue
    if (err) {
      return err;
    } else {
      if (Math.abs(difference) >= 0.25) {
        sendmail({
          from: 'stock_advisor@enigma.com',
          to: 'mohammad.nayer@city.ac.uk',
          subject: 'Project Enigma - Fluctuation Alert',
          html: `STOCK FLUCTUATION ALERT FOR ${tag} AT ${value}`
        }, function (err, reply) {
          console.log(err && err.stack)
          console.dir(reply)
        })
      }
      return;
    }
  });
```

5.3.3.3 Stock Price Retrieval for Tags with Allocated Stock Labels

As mentioned in the sections above, when a user merges a stock label onto a tag in the database, the stock prices for that label are also retrieved at 1 min intervals. This process was intended to be automated, however since the stock labels are on most occasions different to the company names, some sort of manual intervention was required to get these labels onto the system. Once these labels are added, however, the process is then automated for maintaining its values over time.

For this, the AlphaVantage node module (which calls the AlphaVantage API) is used in order to retrieve stock prices. Sample JSON returned looks like this:

```
{
  "Meta Data": {
      "1. Information": "Intraday (1min) open, high, low, close prices and volume",
      "2. Symbol": "MSFT",
      "3. Last Refreshed": "2019-04-16 16:00:00",
      "4. Interval": "1min",
      "5. Output Size": "Compact",
      "6. Time Zone": "US/Eastern"
  },
  "Time Series (1min)": {
      "2019-04-16 16:00:00": {
          "1. open": "120.7000",
          "2. high": "120.8000",
          "3. low": "120.7000",
          "4. close": "120.7900",
          "5. volume": "211572"
      },
      "2019-04-16 15:59:00": {
          "1. open": "120.7200",
          "2. high": "120.7200",
          "3. low": "120.6600",
```


As these records contain different price types, for consistency, the high value was used across all stock prices for every minute. The JSON field was accessed and stored for every label present in the hashtag table, every minute.

Since the stock market AlphaVantage retrieves data from is based in the US, the timestamps needed to be converted for consistency purposes and this was useful in the experiment conducted for carrying out the comparison.

```
function callAlphaAPI(label, res) {
  var key = "Time Series (1min)";
  var valueKey ="2. high";
  alpha.data.intraday(label, 'full', 'json', '1min').then(data => {
    var timeArray = data[key];

    for (time in timeArray) {
      var tempTime = new Date(time);
      tempTime.setHours(tempTime.getHours() + 5);
      var formattedTS = dateFormat(tempTime, "yyyy-mm-dd HH:MM:ss");

      database.addStockRecord(label, timeArray[time][valueKey], formattedTS, (err) => {
        if (err) {
          return err;
        }
        return;
      });
    }
    return res.send("Done");
  });
```

5.4 Docker File

A description of the actions taken during containerisation follows the DockerFile code:

```
FROM node:8

# DOCKERFILE CODE ADAPTED FROM THE NODEJS DOCKER DOCUMENTATION
WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8080/3001
CMD [ "npm", "start" ]

```

- The first line specifies which version of Node to use as the base image of the Docker container on which the application will be built.
- Next, a working directory is made inside the container, and all the node dependencies from package.json are moved into it.
- Npm install is run to install the dependencies for the app.
- The source code for Enigma is copied from the current directory into the root directory of the container.
- Port 8080 is exposed on the container and mapped to port 3001 internally for the Node app.
- Npm start is specified as the executable command which starts the Node app.

5.5 Data Set

The data set (available in the flash drive submitted offline) contains sentiment records over time for various hashtags that were monitored throughout Enigma's running period. Most of these are companies and technology corporations, however there are a few miscellaneous ones included. This data set also includes the structured analysis done against real time stock prices of the tags used in the experiment as described in section 5.6.

5.6 API Documentation

The documentation specifying how to use Enigma as an API can be found in Appendix B1.

5.7 Experiment Results and Analysis

Section 4.6 outlines the experiment conducted. To summarize, 7 tags were monitored every minute on twitter to gather sentiment data. In parallel to this, the stock labels for each of those 7 tags (belonging to the companies – IBM, Google, Tesla, PayPal, Apple, Walmart and Microsoft) were also used to monitor the stock price changes every minute. This data was gathered for 2 consecutive days between 9:00 AM to 4:30 PM EST, which is when the stock market stays open on the weekdays. The times were converted to match and the data collected was then compared, each of the 7 Enigma sentiment values against their 7 respective stock price values. To do this, the data was exported into CSV format and then entered into SPSS in pairs of sets – the stock prices and the enigma values, and then evaluated by using the SPSS Expert Modeler, which is a feature in SPSS used to analyse time series data in order to discover which type of statistical model best describes the time series and how the evaluation was made. It is aimed at attempting to forecast future values in the time series based on previous values. Not all time series datasets are able to do this, and so if a certain significance does not hold, this forecasting cannot be done.

In this report, details as to how statistical time series models behave and what their characteristics are will be avoided, primarily because these are irrelevant and not required in order to decide whether or not the pairs of time series datasets are related. The complete data analysis for all 7 comparisons can be found in the appendix, but since this will be too large to fit within the page limit of the report, only 1 case will be explained, and the steps taken while analysing this data to reach the conclusion will be outlined. A table however will be shown at the end which summarizes all the main findings for all 7 comparisons. Aid was taken from the City University Data Mining module content, prepared by Peter Smith for this section. The material can be found in the appendix.

Since the goal of the analysis was not to make predictions on future stock price changes (even if in some cases this was produced as part of the analysis), a lot of statistical concepts will briefly be described, enough to have some background understanding behind the decisions made.

An interesting observation to note – Walmart had much fewer tweets over the time period compared to other tags, so there were more constant values in the Enigma graph compared to its stock graph. This was an accidental benefit as this allowed more variety of data types to be introduced into the analysis.

Some concepts to mention before walking through the analysis of the selected tag:

ARIMA (AutoRegressive Integrated Moving Average) models are a type of statistical model used to describe time series data as a measure of 3 components. These are auto regression (p), integration (q) and moving average (d). The various ARIMA models take the form of Arima(p,q,d) where p,q and d can be numbers. Each model describes a certain type of model.

Auto Correlation Functions (ACF) are a measure of how a time series may relate with a delayed or “lagged” copy of itself. PACF or Partial Auto Correlation Functions are a variant of ACF and these are both used to help evaluate trends within difference orders of the same time series. These can also be used to create correlograms, which are visual representations of the time series correlates with its own lagged version.

The Ljung-Box Q test is a significance test and a way of measuring whether or not the next point in the time series is affected purely by the one before it. This is used primarily for forecasting and prediction.

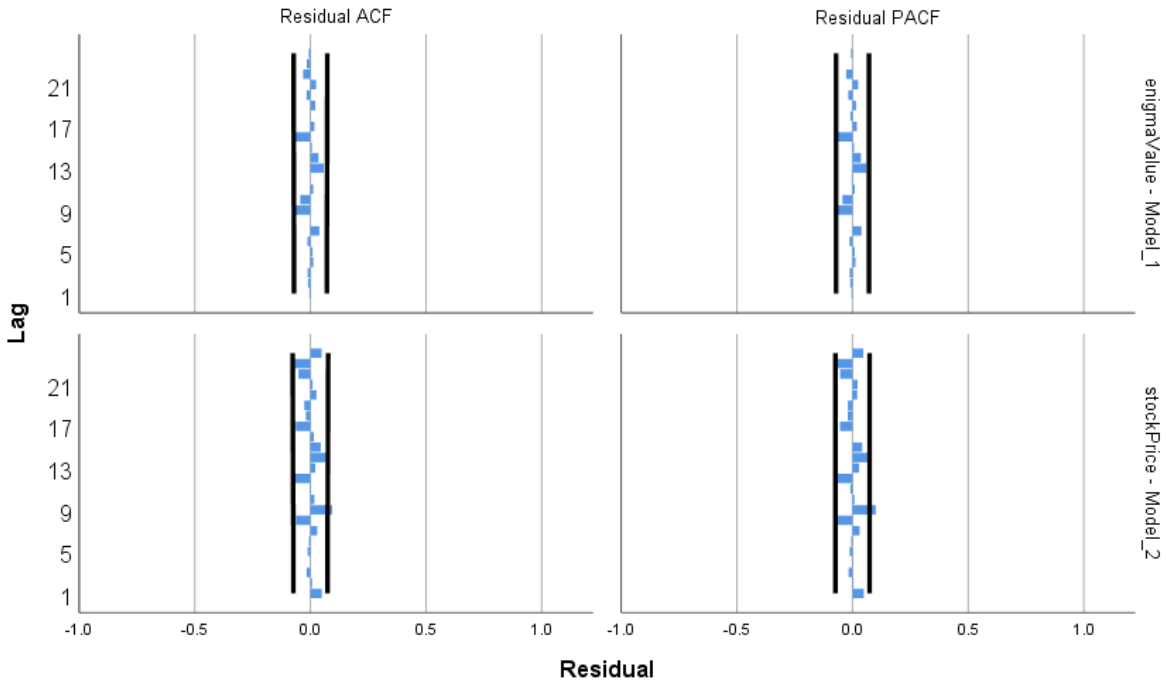
In addition to these, the time series may be transformed based on natural log or a difference value to better obtain or observe a trend.

All of these characteristics were used to reach a decision on whether or not the 2 time series data sets could be related.

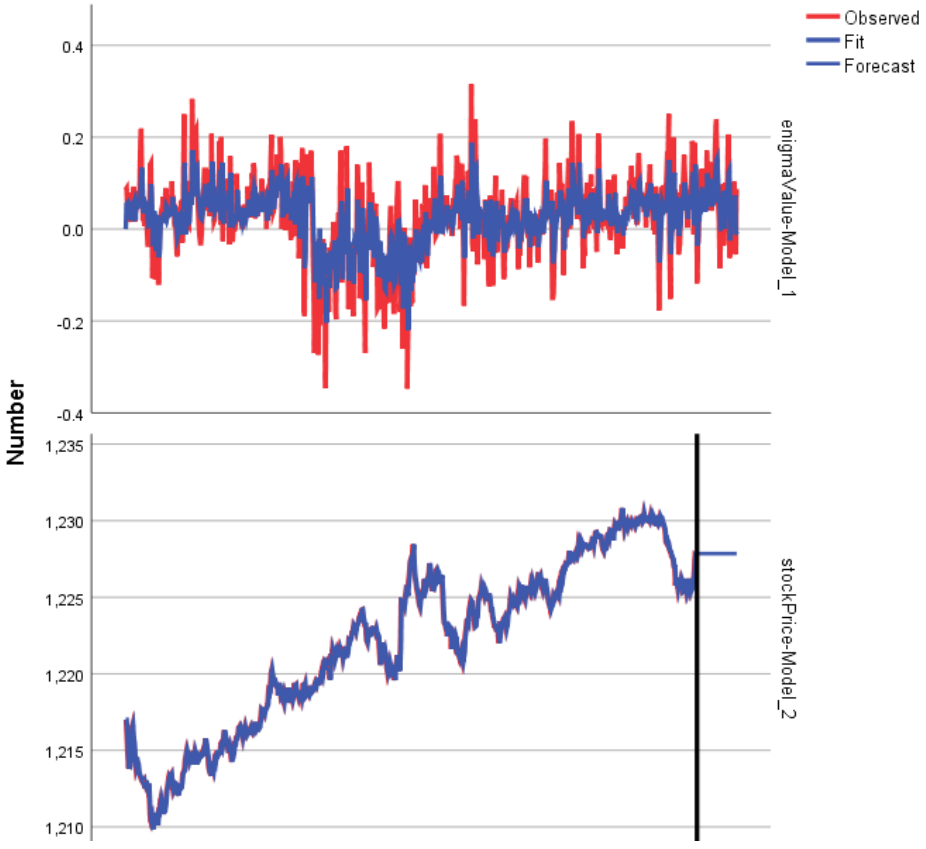
The comparison which will be discussed in the main body of the report is the one done on Google. To start off with the analysis output, SPSS has provided the Model types for both data sets. On this occasion, the enigma time series is allocated the Arima (1,0,3) model and the Stock Price data set is allocated a “Simple” model type. This tells us that the models for both datasets are not the same.

Next, the Ljung-Box significance value reached for the enigma dataset is 0.446, while it is 0.105 for the stock price dataset. This tells us that the stock price dataset has a nature similar to a data set with points predictable from the previous values collectively as opposed to the immediate previous value. This in turn allowed a forecast to be made for the stock price dataset, but no forecast could be made for the enigma dataset. This also points out another difference in the datasets.

Residual ACF and PACF for both datasets did not display and useful trends.



Finally, there is a clear upwards linear trend for the stock price dataset, however no such trend can be observed for the enigma data set. The appropriateness of the model fits (especially on the stock price one) also show us that there is less room for misjudgement.



Below is a table summarising the findings from the comparison for each of the 7 data set pairs:

Tag/Stock	Stock Prices Model Outcome / LjungBox Sig. Value	Enigma Values Model Outcome / LjungBox Sig. Value	Stock Price / Enigma Value Forecastable?	PACF / ACF trends observed	Fit comparison	Decision
Walmart	Arima(0,1,5) / 0.019	Simple / 0.22	Yes / No	Downward trend for stock only	Almost perfect for both	No relation
Google	Simple / 0.105	Arima(1,0,3) / 0.446	Yes / No	None	Perfect for stock, slightly improper for enigma	No relation
Microsoft	Arima(0,1,5) / 0.114	Arima(1,0,0) / 0.788	Yes / No	None	Almost perfect for stock only	No relation
Apple	Arima(0,1,1) / 0.74	Arima(0,0,2) / 0.45	No / No	Upward trend for enigma only	Almost perfect for stock only	No relation
IBM	Simple / 0.098	Arima(0,1,5) / 0.092	Yes / No	Normal distribution for enigma only	Almost perfect for both	No relation
PayPal	Arima(1,1,0) / 0.150	Arima(1,0,1) / 0.070	Yes / Yes	Sine-wave for stock only	Almost perfect for stock only	No relation
Tesla	Arima(0,1,3) / 0.08	Arima(2,0,0) / 0.572	Yes / No	None	Almost Perfect for both	No relation

Some interesting observations made were in almost all cases, the stock prices could be forecasted, whereas in almost all cases no forecasts could be made for enigma values. There were also a few enigma value instances with very high significance values. All of the cases resulted in different models fitting the 2 datasets. With this evaluation, it can be concluded that there is no evidence beyond doubt of any relationship between the 2 datasets with the data that was used. Further discussion on this and the project as a whole will be made in Chapter 6.

Chapter 6: Conclusions and Discussion

6.1 Discussion of Results from Data Analysis

A few observations were made throughout the development stages of Enigma, which are believed to perhaps better explain why the results from the data analysis turned out the way they did.

First of all, it was very quickly made clear that by the time users have started to tweet about a product or service which expects to influence the stock price of the company, the stock prices have already begun to fluctuate well before this. The reason for this is because due to research and awareness, services such as Bloomberg have already provided direct and quick data to shareholders using Bloomberg as a service to inform them of the expected changes. On a few occasions, a few major shareholders will plan a sell/buy in coordination by some cooperative means. This is especially true for top level executives within the company, who may withhold the information before releasing it. Whatever the reasons, these are some factors which are spoken about amongst individuals active in the financial space.

Although the results were clearly in favour of the decision that there is no relation between the data Enigma had generated and the actual stock prices, I believe this may be because of the way sentiment analysis had been done and stored and also due to a time scale factor.

While the number of points and values used to conduct the experiment were in theory sufficient, it may be that specifying a finer granularity in selecting time periods was a reason as to why no correlation could be made. Perhaps a larger time period over a longer amount of time would have yielded more promising results? Since this has not been tested within this report and there was not really any critical evaluation done as to why one should have been chosen over the other, I feel there is potential for discoveries to be made by varying this component of the experiment.

Due to the nature of the stock market (and this is something Enigma targeted originally) it is usually due to events such as acquisitions, bankruptcies, security breaches etc. that stock price fluctuations occur dramatically over an unexpected time suddenly. Since none of these took place during the course of the experiment, this could not be testing, and the hope was that even if no direct relationship could be made, at least a relationship would emerge when a trigger event would set off waves on both social media and the stock markets.

Addressing the point made about how the sentiment analysis was carried out and the value stored, although the sentiment value being a product of statement polarity and subjectivity

had sound reasoning behind it, this was purely experimental and was to be tested. Perhaps another relationship between the 2 variables could be found and used, or perhaps a completely different approach altogether can be used to evaluate sentiment.

It was also noticed from the data analysis that much of Enigma's time series data seemed to be "noisy". Where the stock price data had low significance values and most of these could fit a statistical model almost perfectly, this was not the case with basically all of the enigma data. Perhaps the reason for the extremely mean centred data was the use of the average of all the tweets to represent the sentiment for all tweets in a given minute. Since positive and negative tweets are in theory made all the time, collecting the sentiment from 10 tweets and averaging this might as well have destroyed the purpose of data collection. Another approach could have been to retrieve fewer than 10 tweets every minute and instead of averaging them, use the actual values. This would preserve the original sentiments of all tweets and reduce the mean centred nature of the data. However, this could not be tested due to API limitations.

Due to these reasons, what this experiment concludes is not that no relation exists between sentiment from tweets on twitter about a company and its stock price, rather the conclusion is that if a relationship exists, this was not the appropriate way to find and measure it. The approach used to identify this relationship was very one dimensional and making the changes mentioned in this section, or reconducting the experiment with various values of time period, tweet number and sentiment calculation would prove to be a much more inclusive and reliable experiment.

6.2 Project Problem and Requirements

Taking look back at the main objective and all the subobjectives as specified in the PDD and chapter 1, and as mentioned in chapter 1, these have all been met. Although the relationship could not be proved by this approach, Enigma does do everything which was said it would do. The only major change to this is that Enigma's GUI was excluded with the intent of it being used as an API which has a sound reason behind it. Producing a very limited GUI for it when this feature can be better exploited and personally developed (if at all it is even needed) based on preferences by its users will not only allow for better user experience but will also enable and encourage them to further extend the functionality it provides and make it more useful.

6.3 Problems Encountered

There were a few occasions on which research, reasoning and evaluation were made extra use of in order to solve, and these did prove to be challenging issues as they involved large restructures or changes to the way the over project came together. These have been discussed in detail in this report before and these were:

- The issue of the Perceptron based classifier which caused the whole method of which values were to be used, storing the values, how they were processed and what the output was. Thankfully a quick restructure allowed for this issue to be solved and a much better final outcome to be reached.
- The inability to automate the stock price retrieval process. Although this could not be solved, a suitable workaround was designed which required as little manual interaction as possible to get the job done.

6.4 Personal Evaluation

Overall, I believe Enigma allowed me to really challenge myself in several dimensions, from really exerting to document requirements and changes as much as possible, as well as technologically by allowed me to learn and develop using a new technology stack. I believe this was the most beneficial outcome from this project. It has enabled me to have a better understanding on how to approach tasks for the future methodically, architecturally and technically. The delivery of a completed project with an outcome only seems to be a bonus in this context.

6.5 Future Work

As mentioned in 6.1, there are various ways the experiment can be reconducted by making the necessary changes (and more) to either try and establish a connection between the 2 attributes being measured, or even to change the attributes as a whole.

Machine Learning and AI can be applied to the Enigma model which will constantly be evaluated against actual stock prices and tweak the way Enigma does its own sentiment evaluation or the type of relation it develops between the 2.

As mentioned, Enigma does not have a GUI, and this can be added very simplistically to bring its core functionality into 1 place for users not intending to use it as an API.

Because of the type of project Enigma is, really anything can be made better about it across all dimensions. This includes data analysis and experimentation as well as the type of data it processes and even what the goal of the processing is. Its data sources can be extended to include multiple social media and even beyond instead of just twitter. Using better APIs, the frequency and type of information processing and gathering can be modified. Finally, instead of using predefined APIs, a fully custom one can be created just for it wherein its behaviour can be fully personalised for the most specific experiments and features.

Reference List

1) Dingsøyr, T., Dybå, T. and Moe, N. (2010). *Agile software development*. Berlin: Springer.
2) Sommerville, I. (2011). *Software engineering*. Boston: Pearson.
3) Pozzi, F., Fersini, E., Messina, E. and Liu, B. (n.d.). *Sentiment analysis in social networks*.
4) Nlp.stanford.edu. (2019). *Tokenization*. [online] Available at: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html [Accessed 15 Apr. 2019].
5) Lu, R. and Yang, Q. (2012). Trend Analysis of News Topics on Twitter. *International Journal of Machine Learning and Computing*, pp.327-332.
6) Bagheri, H. and Islam, M. (2017). *Sentiment Analysis of Twitter Data*. [online] Arxiv.org. Available at: https://arxiv.org/ftp/arxiv/papers/1711/1711.10377.pdf [Accessed 16 Apr. 2019].
7) Yaffee, R. and McGee, M. (2009). *Introduction to time series analysis and forecasting*. San Diego [etc.]: Academic Press.
8) Metcalfe, A. and Cowpertwait, P. (2009). *Introductory Time Series with R*. New York, NY: Springer-Verlag New York.
9) Jones, M. (n.d.). *Python for Complete Beginners*.
10) Brown, E. (2014). *Web development with Node and Express*. Sebastopol, CA: O'Reilly Media.

Appendices

Appendix A: Project Definition Document

Project Definition Document

Degree Programme: BSc (Hons) Computer Science (with Placement Year)

Date: 31-October-2018, Academic Year 2018-2019

Project Code Name: Enigma

Project Title: Natural Language Processing and Sentiment Analysis on Tweets to Predict Influence of Tweets on Stock Prices

This is a Design and Build Project.

Supervisor: Jacob Howe

Project Proposed By: [redacted]

Arrangements for Proprietary Interests: This is not a client-based project, so these do not apply. There are no such arrangements.

Promises Made to Secure Acceptance of Project: None. The project has already been agreed to and accepted by my supervisor and added into the Project Allocation Tool.

Proposal Word Count: 1488

[redacted]

[redacted]

[redacted]

Project Definition Document: Proposal

Problem to be Solved

This project (Enigma) aims to develop a tool by exploring the opportunity that has arisen from advances in information technologies and will take advantage of software maintenance and running mechanisms currently being used in the industry. When completed successfully, Enigma will enable interested parties to not only extract meaningfulness from data that has already existed but will also convert this data into actionable insights.

Twitter is one such source of data. About 500 million tweets are made each day, and each of these conveys a piece of information (not all useful) from a certain individual or organisation which may be used. A subset of all these tweets contains tweets from consumers, clients, partners, collaborators and academics regarding a certain organisation, product or service. Enigma will employ the use of natural language processing on these tweets based on certain search criteria, and by carrying out sentiment analysis, will be able to display the overall sentiment visually. Near completion, Enigma will be able to make recommendations and predictions regarding large stock price fluctuations. Essentially, the question being answered is "How can tweets be used to reliably predict the influence of people's opinions (on a certain product/service) on the stock price of the product vendor/service provider?"

Please see the references section at the end of the proposal section for references to existing relevant work. It should be clear how these relate to my project based on the above description.

Project Objectives

Main Objective: Enigma shall allow users to input search criteria (e.g., company/product names) into a tool and be able to make recommendations about the predicted imminent changes in stock price for that company (if there are any).

The sub-objectives which must be met to achieve this are:

1) Enigma shall have a UI which will allow the user to input search parameters
   Test: UI successfully accepts and transfers user input to backend
2) Enigma shall have a server to serve client requests as mentioned above
   Test: Backend processes user input and sends back a response
3) Enigma shall carry out Natural Language Processing on tweets which match user's criteria
   Test: Raw data is converted into a form which can be interpreted by sentiment analyser
4) Enigma shall extract overall sentiment (with confidence rating) about mentioned tweets and display this to the user visually on the UI
   Test: Sentiment Analysis is accurate (can be tested by dummy data) and successfully returns a useful value

After these basic objectives have been met, some additional features will be added to enhance Enigma's functionality:

5) Enigma shall make predictions (using AI / machine learning) on stock price fluctuations
   Test: Accuracy of predictions can be matched (and trained) by using historical data

6) Enigma shall take advantage of containerisation (Docker) for running an isolated environment and to make maintenance and updating easy and clean
   Test: The required Docker containers are spun up and run and each reflect the behaviour of the respective parts of Enigma that they will cover
7) Enigma shall exist live on the cloud, continuously monitoring certain keywords for sudden sentiment shifts, and alert the user if a large fluctuation occurs
   Test: Emails are triggered when a fluctuation (over a pre-determined value) is observed on a particular keyword

Project Beneficiaries

Individuals or organisations which may benefit from Enigma's completion:

- Further developers interested in Natural Language Processing and Sentiment Analysis which is carried out on vast sources of raw data such as Twitter, and how AI develops on this over time
- Researchers and individuals in statistics/finance who are interested in making relationships between tweet content, tweet frequency and stock price fluctuation within short time windows. People in marketing may see the impact of an exceptionally good or bad demo about a product/service on Twitter
- Students and individuals of all ages interested in technology and finance who can understand what can be achieved by applying computer science to real world situations

Work Plan

I intend to use the Agile software development methodology to deliver Enigma. This means I will maintain a "do and test as you go" approach as opposed to relying on heavy requirements documentation and planning before beginning development. This does not mean however, that there will be no documentation. Every action taken to meet a requirement and the analysis behind it will be documented. I intend to use 2-week sprints which started from the second half of October, and will run up to the end of March. Each sprint will have a deliverable. Below is a list of tasks I will be completing over time. Below, I have addressed each of the sub objectives as mentioned in "Project Objectives" and turned them into components/deliverables for Enigma:

Sprint 1, 2 (mid Oct – mid Nov): Understand the technologies being used (as these are new to me) for the UI and the server/backend and bring them together as a working web app. Install all necessary software, packages and plugins etc. For the backend, I will be using the

language NodeJS with the Express framework. For the UI, I will be using ReactJS with HTML and CSS.

Sprint 3, 4 (mid Nov – mid Dec): Explore Twitter API, Explore and implement Natural Language Processing techniques and procedures. Be able to retrieve tweets on a keyword (hash tag) and have basic NLP running on them.

Sprint 5, 6, 7 (mid Dec – end of Jan): Sentiment analysis (SA) is the main component of Enigma. In sprint 5, I will be researching and exploring SA techniques and procedures and attempting to implement these until sprint 6. Sprint 6 will also focus on checking how accurate the SA is by checking it against dummy data and improving the SA procedure. Sprint 7 may carry over some work from sprint 6, and the rest of sprint 7 will be used to containerise the application using Docker and place the application on the cloud, so it is accessible. All this will be happening with exam preparation, so will also take relatively more time.

Sprint 8, 9, 10 (Feb – mid March): Another large component - explore machine learning, neural networks, genetic algorithms and/or frameworks (such as TensorFlow) to use AI in to further improve sentiment analysis and use this to be able to make predictions. Prediction accuracy can also be checked based on real values and can be used to train the model.

Sprint 11 (mid March – end of March): Implement the additional feature of alerting the user on live large fluctuations on predefined keywords. This can be skipped, and these 2 weeks can be used as flexibility if any previous sprint(s) over run.

Sprint	Deliverable	Date
1	Backend server-side logic	15 Oct 18 – 31 Oct 18
2	Front end User Interface	1 Nov 18 – 15 Nov 18
3	Explore Twitter API, Research NLP procedures	16 Nov 18 – 30 Nov 18
4	Implement basic NLP and have it working on tweets	1 Dec 18 – 15 Dec 18
5	Research SA procedures, begin implementation	16 Dec 18 – 31 Dec 18
6	Continue SA implementation, test and improve against dummy data	1 Jan 18 – 15 Jan 18
7	Containerize application, move to cloud	16 Jan 18 – 31 Jan 18
ENIGMA BASIC REQUIREMENTS MET		
8	Explore AI, machine learning, neural networks	1 Feb 18 – 15 Feb 18
9	Begin integrating findings from sprint 8 to improve SA, check against dummy data and to train the model	16 Feb 18 – 28 Feb 18
10	Use findings from sprint 8 to enable prediction feature and repeat items from sprint 9 on prediction	1 Mar 18 – 15 Mar 18
11	Implement feature to alert user on large fluctuations by continuously checking live tweets	16 Mar 18 – 31 Mar 18

As of right now, I am only 100% sure about the implementation technology/procedure for sprint 1 and 2. As the sprints come along, I have accounted for research/exploration time to pick a certain technology/procedure and will decide when I start the respective component. This work plan has accounted for most variables and can be considered feasible. All documentation/outputs will follow the “do as you go” approach and are part of each sprint. Enigma involves a lot of research and learning and this has also been considered.

Please Note: I have not included a Gantt chart as I feel this is will be redundant for my approach. None of the sprints are intended to overlap, and in the Gantt chart, this would show up as sequential in the same way as the list above. At all times during the 11 sprints, I will be coding, researching, adding new requirements, testing and documenting at the same time. I have instead used the table to outline sprints and their objectives with dates.

Project Risks

These are some main risks that Enigma is associated with, and how they will be minimized or contained if they do become a problem:

1) General hardware/theft /loss issues: Enigma’s development will be maintained in a private GitHub repository, so nothing is lost if this issue arises.

2) General time shortage issues: This includes things like sickness, absence, or over running sprints. I have made it so that sprint 11 can be replaced for extra time, as sprint 11 delivers an extra feature which is not important in the case that this does happen.
3) New technology learning/implementing: As mentioned earlier, Enigma involves a lot of research. In the case I have under estimated how long it will take to learn certain aspects of the project (highly unlikely), I may have to compromise on some of the extra features (such as containerisation and cloud hosting) to make time for this and deliver the minimum viable product.

There is no risk to Enigma's potential users as such. If users consider the predictions as serious and take actions (such as investing) based on its "advice", they risk losing wealth. To avoid this, it will be clear that Enigma is an educational tool and is not intended to encourage users to invest in any way based on its findings.

References

1) Eugene, A. (2018). *Using the Twitter API and NLP to analyze the tweets of different users*. [Online] Towards Data Scientist. Available at: https://towardsdatascience.com/twitter-api-and-nlp-7a386758eb31 [Accessed 30-Oct-18]
2) Stephanie, K. (2016). *An NLP Approach to Analyzing Twitter, Trump, and Profanity*. [Online] Algorithmia. Available at: https://blog.algorithmia.com/nlp-approach-twitter-trump-profanity/ [Accessed 30-Oct-18]
3) Hussain, M. (2017). *Applying NLP to Tweets With Python.* [Online] Dzone. Available at: https://dzone.com/articles/applying-nlp-to-tweets-with-python [Accessed 30-Oct-18]
4) Robinson, S. (2017) *Building a realtime Twitter sentiment dashboard with Firebase and NLP*. [Online] Codeburst. Available at: https://codeburst.io/building-a-realtime-twitter-sentiment-dashboard-with-firebase-and-nlp-7064bb30f5ab [Accessed 30-Oct-18]

Research Ethics Checklist

Ethics Review Form: BSc, MSci, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

Part A: Ethics Checklist. All students must complete this part. The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

Part B: Ethics Proportionate Review Form. Students who have answered "no" to questions 1 – 18 and "yes" to question 19 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in this case. The approval may be provisional: the student may need to seek additional approval from the supervisor as the project progresses.

A.1 If your answer to any of the following questions (1 – 3) is YES, you must apply to an appropriate external ethics committee for approval.		*Delete as appropriate*
1.	Does your project require approval from the National Research Ethics Service (NRES)? For example, because you are recruiting current NHS patients or staff? If you are unsure, please check at http://www.hra.nhs.uk/research-community/before-you-apply/determine-which-review-body-approvals-are-required/.	**No**
2.	Does your project involve participants who are covered by the Mental Capacity Act? If so, you will need approval from an external ethics committee such as NRES or the Social Care Research Ethics Committee http://www.scie.org.uk/research/ethics-committee/.	**No**
3.	Does your project involve participants who are currently under the auspices of the Criminal Justice System? For example, but not limited to, people on remand, prisoners and those on probation? If so, you will need approval from the ethics approval system of the National Offender Management Service.	**No**

A.2 If your answer to any of the following questions (4 – 11) is YES, you must apply to the City University Senate Research Ethics Committee (SREC) for approval (unless you are applying to an external ethics committee).	*Delete as appropriate*

[redacted] [redacted]

4.	Does your project involve participants who are unable to give informed consent? For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf?	**No**
5.	Is there a risk that your project might lead to disclosures from participants concerning their involvement in illegal activities?	**No**
6.	Is there a risk that obscene and or illegal material may need to be accessed for your project (including online content and other material)?	**No**
7.	Does your project involve participants disclosing information about sensitive subjects? For example, but not limited to, health status, sexual behaviour, political behaviour, domestic violence.	**No**
8.	Does your project involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning? (See http://www.fco.gov.uk/en/)	**No**
9.	Does your project involve physically invasive or intrusive procedures? For example, these may include, but are not limited to, electrical stimulation, heat, cold or bruising.	**No**
10.	Does your project involve animals?	**No**
11.	Does your project involve the administration of drugs, placebos or other substances to study participants?	**No**

A.3 If your answer to any of the following questions (12 – 18) is YES, you must submit a full application to the Computer Science Research Ethics Committee (CSREC) for approval (unless you are applying to an external ethics committee or the Senate Research Ethics Committee). Your application may be referred to the Senate Research Ethics Committee.		*Delete as appropriate*
12.	Does your project involve participants who are under the age of 18?	**No**
13.	Does your project involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.	**No**
14.	Does your project involve participants who are recruited because they are staff or students of City University London? For example, students studying on a specific course or module. (If yes, approval is also required from the Head of Department or Programme Director.)	**No**
15.	Does your project involve intentional deception of participants?	**No**

[redacted] [redacted]

16.	Does your project involve participants taking part without their informed consent?	**No**
17.	Does your project pose a risk to participants or other individuals greater than that in normal working life?	**No**
18.	Does your project pose a risk to you, the researcher, greater than that in normal working life?	**No**

A.4 If your answer to the following question (19) is YES and your answer to all questions 1 – 18 is NO, you must complete part B of this form.		
19.	Does your project involve human participants or their identifiable personal data? For example, as interviewees, respondents to a survey or participants in testing.	**No**

Part B: Ethics Proportionate Review Form (DOES NOT APPLY)

If you answered YES to question 19 and NO to all questions 1 – 18, you may use this part of the form to submit an application for a proportionate ethics review of your project. Your project supervisor has delegated authority to review and approve this application.

However, if you cannot provide all the required attachments (see B.3) with your project proposal (e.g. because you have not yet written the consent forms, interview schedules etc), the approval from your supervisor will be provisional. You **must** submit the missing items to your supervisor for approval prior to commencing these parts of your project. Failure to do so may result in you failing the project module.

There may also be circumstances in which your supervisor will ask you to submit a full ethics application to the CSREC, e.g. if your supervisor feels unable to approve your application or if you need an approval letter from the CSREC for an external organisation.

B.1 The following questions (20 – 24) must be answered fully.		*Delete as appropriate*
20.	Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**N/A**
21.	Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**N/A**
22.	When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**N/A**
23.	Will consent be obtained from the participants in your project?	**N/A**

	Consent from participants will be necessary if you plan to involve them in your project or if you plan to use identifiable personal data from existing records. "Identifiable personal data" means data relating to a living person who might be identifiable if the record includes their name, username, student id, DNA, fingerprint, address, etc. *If YES, you must attach drafts of the participant information sheet(s) and consent form(s) that you will use in section B.3 or, in the case of an existing dataset, provide details of how consent has been obtained.* *You must also retain the completed forms for subsequent inspection. Failure to provide the completed consent request forms will result in withdrawal of any earlier ethical approval of your project.*	
24.	Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential? Provide details:	**N/A**

B.2 If the answer to the following question (25) is YES, you must provide details		*Delete as appropriate*
25.	Will the research be conducted in the participant's home or other non-University location? *If **YES**, provide details of how your safety will be ensured:*	**N/A**

B.3 Attachments (these should be provided if applicable):	*Delete as appropriate*
Participant information sheet(s)**	**Not applicable**
Consent form(s)**	**Not applicable**
Questionnaire(s)**	**Not applicable**
Topic guide(s) for interviews and focus groups**	**Not applicable**
Permission from external organisations (e.g. for recruitment of participants)**	**Not applicable**

[redacted] [redacted]

If these items are not available at the time of submitting your project proposal, provisional approval through proportionate review can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. **All such items **must** be seen and approved by your supervisor before the activity for which they are needed starts.

Templates

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. These are available from the links below but you **must** adapt them according to the needs of your project before you submit them for consideration.

Adult information sheet:

http://www.city.ac.uk/__data/assets/word_doc/0018/153441/TEMPLATE-FOR-PARTICIAPNT-INFORMATION-SHEET.doc

Adult consent form:

http://www.city.ac.uk/__data/assets/word_doc/0004/153418/TEMPLATE-FOR-CONSENT-FORM.doc

Further Information

Information about the Computer Science Research Ethics Committee (CSREC) is available at: http://www.city.ac.uk/department-computer-science/research-ethics

Information about the City University Senate Research Ethics Committee is available at: http://www.city.ac.uk/research/research-and-enterprise/research-ethics

Appendix B: Outputs

Appendix B1: API Specification

The following routes are defined and can be made requests using the HTTP GET method. This may be done from a browser or using an HTTP request generator such as Curl or postman, and of course any other application:

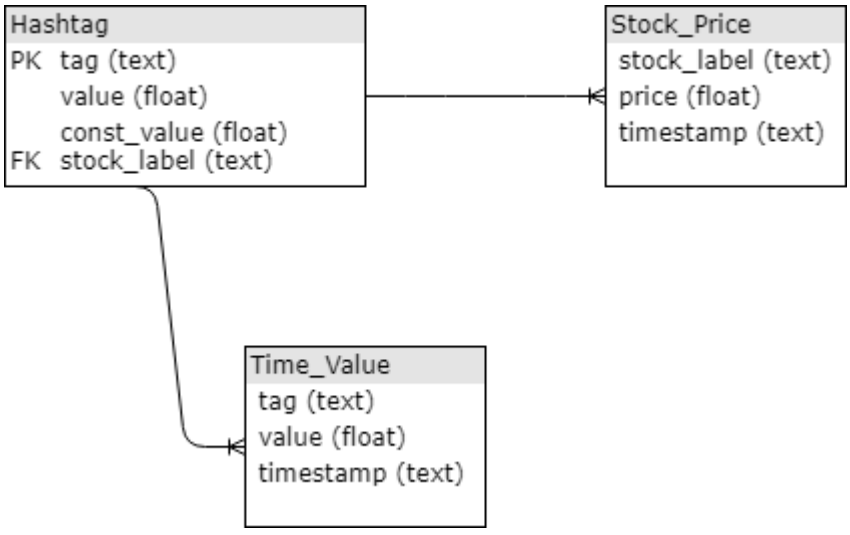
/tag/add – This must be given a “query” parameter, which is the tag needed to be added onto Enigma. The result is that the tag is run through the Python script and the value stored in the database so that it can be updated every specified time period by the Node application.

/tag/view – This must be given a “query” parameter which is the tag that is already on the database, which has been maintained. This plots a graph to show the user with all the value points stored by enigma on that tag over time. A link is returned to the user to see the graph.

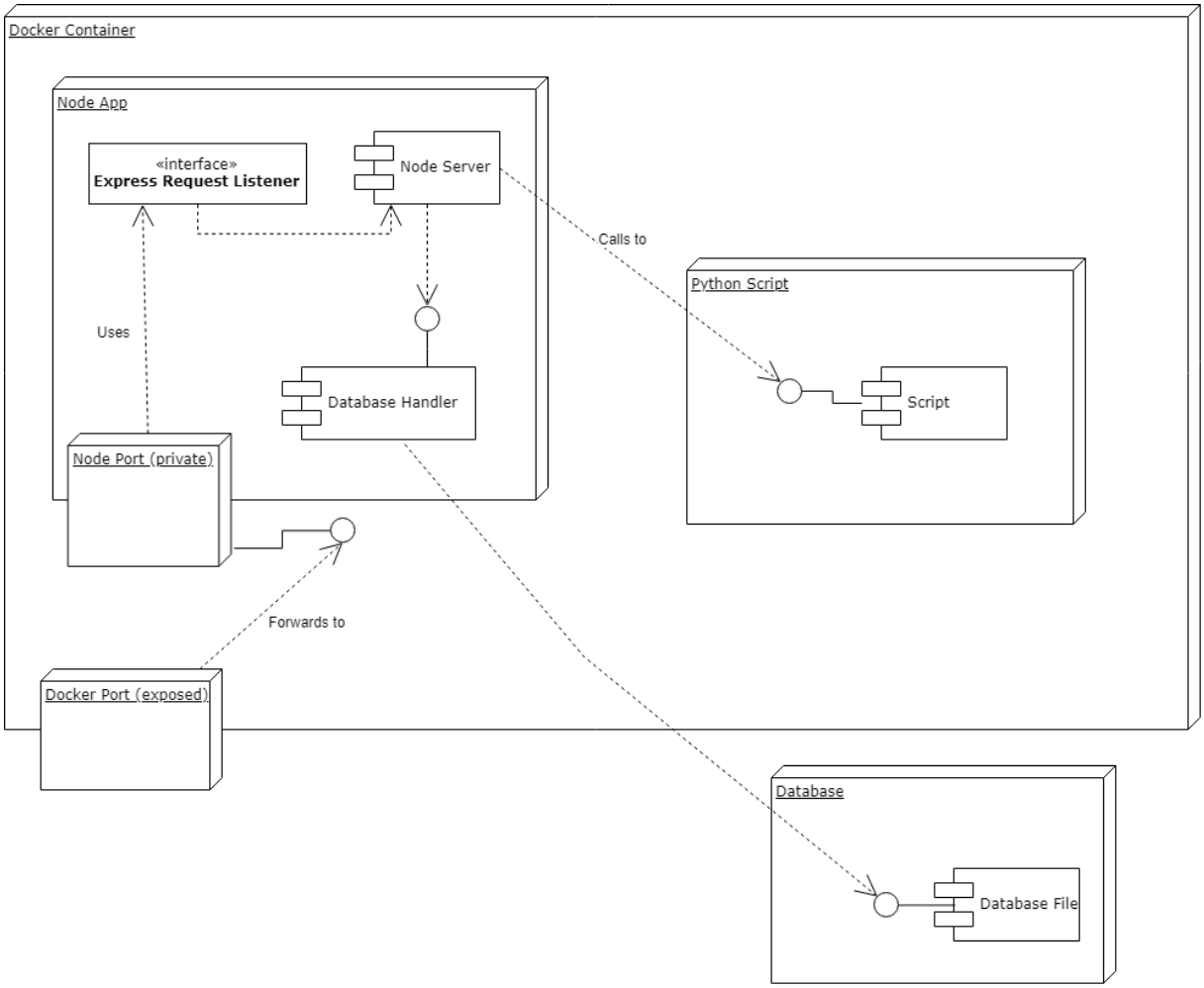
/stock/add – along with a “merge” parameter in the format <TAG>*<STOCK_LABEL>, i.e., two strings with asterisk in between. For example: /stock/add/?merge=Google*GOOG. This allows the user to link a tag in the database with a stock label (found by the user) so that the stock prices for the tag can also be monitored in the same time period for comparing against enigma’s values by the Node application

/stock/view– along with a query parameter which is the stock label which is on the database, whose price has been followed. This plots a graph to show the user with all the price points stored by enigma on that stock label over time. A link is provided to the user to see the graph.

Appendix B2: Entity-Relationship Diagram



Appendix B3: Component Diagram



Appendix B4: Dataset

Appendix B4, the dataset is provided as part of the offline submission.

Appendix B5: Python Script

Please note, the application will not work without the appropriate access tokens and API keys being set.

```
import re

import sys

import tweepy

from tweepy import OAuthHandler

from textblob import TextBlob
```

```
class TwitterClient(object):

    '''
    Class for sentiment analysis of tweets.
    '''

    def __init__(self):
        consumer_key = 'HIDDEN'
        consumer_secret = 'HIDDEN'
        access_token = 'HIDDEN'
        access_token_secret = 'HIDDEN'

        try:
            self.auth = OAuthHandler(consumer_key, consumer_secret)
            self.auth.set_access_token(access_token, access_token_secret)
            self.api = tweepy.API(self.auth)
        except:
            print("Error: Authentication Failed")

    def clean_tweet(self, tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^#0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

    def get_tweet_sentiment_demo(self, tweet):
        analysis = TextBlob(self.clean_tweet(tweet))
        subjectivity = analysis.sentiment.subjectivity
        polarity = analysis.sentiment.polarity
        sentiment_value = polarity * subjectivity

        if sentiment_value > 0.25:
            return 'positive at ' + str(sentiment_value)
```

```
    elif sentiment_value < -0.25:
        return 'negative at ' + str(sentiment_value)
    else:
        return 'neutral at ' + str(sentiment_value)

def get_tweet_sentiment(self, tweet):
    analysis = TextBlob(self.clean_tweet(tweet))
    subjectivity = analysis.sentiment.subjectivity
    polarity = analysis.sentiment.polarity
    sentiment_value = polarity * subjectivity
    return sentiment_value



def get_tweets_demo(self, query, count):
    try:
        fetched_tweets = self.api.search(q = query, count = count)

        for tweet in fetched_tweets:
            parsed_tweet = {}
            parsed_tweet['text'] = tweet.text
            parsed_tweet['sentiment'] = self.get_tweet_sentiment_demo(tweet.text)

            if tweet.retweet_count > 0:

                if parsed_tweet not in tweets:
                    tweets.append(parsed_tweet)
            else:
                tweets.append(parsed_tweet)

        return tweets
```

```
        except tweepy.TweepError as e:
            print("Error : " + str(e))

    def get_tweets(self, query, count):
        tweets = []
        try:
            fetched_tweets = self.api.search(q = query, count = count)

            for tweet in fetched_tweets:
                parsed_tweet = self.get_tweet_sentiment(tweet.text)
                if tweet.retweet_count > 0:
                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)
                else:
                    tweets.append(parsed_tweet)

            return tweets

        except tweepy.TweepError as e:
            print("Error : " + str(e))

def main():
    api = TwitterClient()
    tweets = api.get_tweets(query = sys.argv[1], count = sys.argv[2])
    print(tweets)
    sys.stdout.flush()


if __name__ == "__main__":
    main()
```

Appendix B6: Node Application with DockerFile

To run Enigma, simply visit the directory it is located in a terminal and type “npm install” (making sure node and python are installed on the machine. Once this is done, “npm start” will start the server and requests will be processed on port 3001. **Please note, the application will not work without the appropriate access tokens and API keys being set.**

```
const express = require('express');

const app = express();

const port = 3001;

const spawn = require('child_process').spawn;

const database = require ('./database.js');

var dateFormat = require('dateformat');

const sendmail = require('sendmail')();


const plotlyKey = 'HIDDEN';

const plotly = require('plotly')('USERNAME', plotlyKey);


const alphaVantageAPIKey = 'HIDDEN';

const alpha = require('alphavantage')({ key: alphaVantageAPIKey });


const oneMinInterval = 1*60*1000;

const constValuePeriod = 180; // in minutes


setInterval(autoUpdateValues, oneMinInterval);


app.get('/tag/add', (req, res) => (req.query.query) ? createNewTag(req.query.query, res) :
res.send("Please specify query!"));

app.get('/tag/view', (req, res) => (req.query.query) ? plotGraph(req.query.query, res) :
res.send('Please specify a query!'));

app.get('/stock/view', (req, res) => (req.query.query) ? plotStockGraph(req.query.query, res) :
res.send('Please specify a query!'));

app.get('/stock/add', (req, res) => {


  if (req.query.merge && req.query.merge.includes("*")) {
```

```
    var tag = req.query.merge.split("*")[0];
    var label = req.query.merge.split("*")[1];

    addNewStock(tag, label, res);
    res.send("Tag merged with label successfully");
  } else {
    res.send('Please specify both values in the format merge=tag*label');
  }
  return;
});

app.get('/stock/view', (req, res) => (req.query.query) ? plotStockGraph(req.query.query, res) :
res.send('Please specify a query!'));

function callScript(query, res) {
  console.log("Script called");
  query = `#${query}`;
  const pythonProcess = spawn('python',["./twitter.py", query, 10]);
  pythonProcess.stdout.on('data', (data) => {
    res.send(`Here are tweet sentiments for ${query}:${data}`);
  });
}

function autoUpdateValues() {
  var tagsArray = [];
  database.getTagNames((err, tagRows) => {
    if (err) {
      return err;
    } else if (tagRows) {
      for(k = 0; k < tagRows.length; k++) {
        tagsArray.push(tagRows[k].tag);
      }
```

```
step(0, 0);
function step(x, constPeriod) {
  if (x < tagsArray.length) {
    var query = `#${tagsArray[x]}`;
    const pythonProcess = spawn('python',["./twitter.py", query, 10]);
    pythonProcess.stdout.on('data', (data) => {

      if (data.toString() && data.toString().split("[") && data.toString().split("[")[1].split("]")) {
        var output = data.toString().split("[")[1].split("]")[0];
        var valuesArray = JSON.parse("[" + output + "]");
        var sum = 0;
        for(m = 0; m < valuesArray.length; m++) {
          sum += valuesArray[m];
        }
        var average = sum / valuesArray.length;
        database.updateTagValue(query.split("#")[1], average, (err) => {
          if(err) {
            return err;
          }
        });
        if (constPeriod >= constValuePeriod) {
          database.updateTagConstValue(query.split("#")[1], average, (err) => {
            if(err) {
              return err;
            }
          });
          if ((constPeriod - constValuePeriod) > tagsArray.length) {
            constPeriod = 0;
          }
        }
        evaluateTagDifference(query.split("#")[1]);
      }
```

```
        });
        step(x + 1, constPeriod + 1);
      }
    }
  }
 });
}


function callAlphaAPI(label, res) {
 var key = "Time Series (1min)";
 var valueKey ="2. high";
 alpha.data.intraday(label, 'full', 'json', '1min').then(data => {
  var timeArray = data[key];

  for (time in timeArray) {
   var tempTime = new Date(time);
   tempTime.setHours(tempTime.getHours() + 5);
   var formattedTS = dateFormat(tempTime, "yyyy-mm-dd HH:MM:ss");

   database.addStockRecord(label, timeArray[time][valueKey], formattedTS, (err) => {
    if (err) {
     return err;
    }
    return;
   });
  }
  return res.send("Done");
 });
}
```

```
function addNewStock(tag, label, res) {
  database.checkIfTagExists(tag, (err, answer) => {
    if (err) {
      return err;
    } else if (answer === false) {
      console.log(`Tag ${tag} does not exist!`);
    } else if (answer === true) {
      database.updateHastagWithStockLabel(tag, label, (err) => {
        if(err) {
          return err;
        }
      });
    }
  });
}

function plotGraph(tag, res) {
  var arrayOfValues = [];
  var arrayOfTimes = [];
  database.checkIfTagExists(tag, (err, answer) => {
    if (err) {
      return err;
    } else if (answer === false) {
      console.log("Error, tag does not exist");
      return;
    } else if (answer === true) {
      database.retrieveTagValueRecords(tag, (err, valueRecords) => {
        if (err){
          return err;
        } else if (valueRecords) {
          for (i = 0; i < valueRecords.length; i++) {
```

```
          arrayOfValues.push(valueRecords[i].value);
        }

        database.retrieveTagTimeRecords(tag, (err, timeRecords) => {
          if (err){
            return err;
          } else if (timeRecords) {
            for (j = 0; j < timeRecords.length; j++) {
              arrayOfTimes.push(timeRecords[j].timestamp);
            }

            var data = [
              {
                x: arrayOfTimes,
                y: arrayOfValues,
                type: "line"
              }
            ];
            var graphOptions = {filename: "date-axes", fileopt: "overwrite"};
            plotly.plot(data, graphOptions, function (err, msg) {
              console.log(msg);
            });
          }
        });
      }
    });
  }
 });
 return res.status(200);
}

function plotStockGraph(label, res) {
```

```
var arrayOfValues = [];
var arrayOfTimes = [];
database.checkIfLabelExists(label, (err, answer) => {
  if (err) {
    return err;
  } else if (answer === false) {
    console.log("Error, label does not exist");
    return;
  } else if (answer === true) {
    database.retrieveStockPriceRecords(label, (err, priceRecords) => {
      if (err){
        return err;
      } else if (priceRecords) {
        for (i = 0; i < priceRecords.length; i++) {
          arrayOfValues.push(priceRecords[i].price);
        }

        database.retrieveStockTimeRecords(label, (err, timeRecords) => {
          if (err){
            return err;
          } else if (timeRecords) {
            for (j = 0; j < timeRecords.length; j++) {
              arrayOfTimes.push(timeRecords[j].timestamp);
            }

            var data = [
              {
                x: arrayOfTimes,
                y: arrayOfValues,
                type: "line"
              }
            ];
```

```
            var graphOptions = {filename: "date-axes", fileopt: "overwrite"};
            plotly.plot(data, graphOptions, function (err, msg) {
              console.log(msg);
            });
          }
        });
      }
    });
  }
});
return res.status(200);
}

function createNewTag(tag, res) {
  database.checkIfTagExists(tag, (err, answer) => {
    if (err) {
      return err;
    } else if (answer === false) {
      var query = `#${tag}`;
      var pythonProcess = spawn('python',["./twitter.py", tag, 10]);
      pythonProcess.stdout.on('data', (data) => {
        var output = data.toString().split("[")[1].split("]")[0];
        var valuesArray = JSON.parse("[" + output + "]");
        var sum = 0;
        for(m = 0; m < valuesArray.length; m++) {
          sum += valuesArray[m];
        }
        var average = sum / valuesArray.length;
        database.addNewTag(query.split("#")[1], average, (err) => {
          if(err) {
            return err;
          }
```

```
      });
    });
    res.send("Tag successfully added");
  } else if (answer === true) {
    res.send("Tag already exists!");
    return;
  }
 });
}

function evaluateTagDifference(tag) {
 return database.calculateTagDifference(tag, (err, difference) => {
  if (err) {
    return err;
  } else {
    if (Math.abs(difference) >= 0.25) {
      sendmail({
        from: '[redacted]
        to: [redacted]
        subject: 'Project Enigma - Fluctuation Alert',
        html: `STOCK FLUCTUATION ALERT FOR ${tag} AT ${value}`
      }, function (err, reply) {
        console.log(err && err.stack)
        console.dir(reply)
      })
    }
    return;
  }
 });
}

app.listen(port, () => console.log(`App listening on port ${port}...`));
```

```
/**************************************************************************************************************
*****************

This file acts as a database handler and contains all functions which deal with the database to make transactions

and to carry out small maniupulations on retreived data inorder to be used by other parts of the application

**************************************************************************************************************
*****************/

const fs = require('fs');

const path = require('path');

const sqlite3 = new require('sqlite3').verbose();

var dateFormat = require('dateformat');


let databasePath = __dirname + '/database.db';


// Check and synchronize database with application at runtime

const databaseDirPath = path.dirname(databasePath);

if (!fs.existsSync(databaseDirPath)) {

  fs.mkdirSync(databaseDirPath);

}


const db = new sqlite3.Database(databasePath);


// Create all tables in the database if they're not already created

function initialiseDatabase() {

  db.serialize(() => {

    db.run('CREATE TABLE IF NOT EXISTS Hashtag (tag TEXT UNIQUE PRIMARY KEY NOT NULL, value INT NOT NULL' +

          ',constant_value INT NOT NULL, stock_label TEXT);');

    db.run('CREATE TABLE IF NOT EXISTS Time_Value (tag TEXT NOT NULL, value INT NOT NULL, timestamp text NOT NULL);');

    db.run('CREATE TABLE IF NOT EXISTS Stock_Price (stock_label TEXT NOT NULL, price INT NOT NULL, timestamp text NOT NULL);');

  });
```

```
}

initialiseDatabase();

db.run('PRAGMA foreign_keys = ON;');

/*****************************************************************************
 * DATABASE MANIPULATION AND INTERACTION FUNCTIONS
 *****************************************************************************/

function updateTagValue(tag, value, callback) {
  db.run(`UPDATE Hashtag SET value = ? WHERE tag = ? COLLATE NOCASE;`, value, tag,
(err) => {
    if (err) {
      return callback(err);
    } else {
    return callback(null);
  }
  });
  addTagRecord(tag, value, (err) => {
    if (err) {
      return err;
    }
    return;
  });
}

function updateTagConstValue(tag, value, callback) {
  db.run(`UPDATE Hashtag SET constant_value = ? WHERE tag = ? COLLATE NOCASE;`,
value, tag, (err) => {
    if (err) {
      return callback(err);
    } else {
```

```
    return callback(null);
  }
  });
}

function updateHastagWithStockLabel(tag, label, callback) {
  db.run(`UPDATE Hashtag SET stock_label = ? WHERE tag = ? COLLATE NOCASE;`, label, tag, (err) => {
    if (err) {
      return callback(err);
    } else {
    return callback(null);
  }
  });
}

function checkIfTagExists(tag, callback) {
  db.all(`SELECT tag FROM Hashtag WHERE tag = ? COLLATE NOCASE;`, tag, (err, response) => {
    if (err) {
      return callback(err, false);
    } else if (response && response[0] !== undefined) {
      return callback(null, true);
  }
    return callback(null, false);
  });
}

function checkIfLabelExists(label, callback) {
  db.all(`SELECT stock_label FROM Stock_Price WHERE stock_label = ? COLLATE NOCASE;`, label, (err, response) => {
    if (err) {
      return callback(err, false);
```

```
    } else if (response && response[0] !== undefined) {
      return callback(null, true);
  }
    return callback(null, false);
  });
}

function addNewTag(tag, value, callback) {
  db.run(`INSERT INTO Hashtag(tag, value, constant_value) VALUES (?, ?, 0);`, tag, value,
(err) => {
    if (err) {
      return callback(err);
    }
    return callback(null);
  });
  addTagRecord(tag, value, (err) => {
    if (err) {
      return err;
    }
    return;
  });
}

function addTagRecord(tag, value, callback) {
  var timestamp = new Date();
  var formattedTS = dateFormat(timestamp, "yyyy-mm-dd HH:MM:ss");

  db.run(`INSERT INTO Time_Value(tag, value, timestamp) VALUES (?, ?, ?);`, tag, value,
formattedTS, (err) => {
    if (err) {
      return callback(err);
    }
    return callback(null);
```

```
  });
}



function addStockRecord(label, price, timestamp, callback) {


  db.run(`INSERT INTO Stock_Price(stock_label, price, timestamp) VALUES (?, ?, ?);`, label,
price, timestamp, (err) => {
    if (err) {
      return callback(err);
    }
    return callback(null);
  });
}


function calculateTagDifference(tag, callback) {
  checkIfTagExists(tag, (err, answer) => {
    if (err) {
      return err;
    } else if (answer === true) {
      db.all(`SELECT constant_value, value FROM Hashtag WHERE tag = ? COLLATE
NOCASE;`, tag, (err, row) => {
        if (err) {
          return callback(err, null);
        } else {
          let difference = row[0].value - row[0].constant_value;
          return callback(null, difference);
        }
      });
    } else if (answer === false) {
      console.log("Error, tag does not exist!");
    }
  });
```

[redacted] [redacted]

```
}


function retrieveTagValueRecords(tag, callback){
  db.all(`SELECT value FROM Time_Value WHERE tag = ? AND timestamp >= "2019-04-16" COLLATE NOCASE;`, tag, (err, rows) => {
    if (err) {
      return callback(err, null);
    } else {
      return callback(null, rows);
    }
    return callback(err, null);
  });
}


function retrieveStockPriceRecords(label, callback){
  db.all(`SELECT price FROM (SELECT DISTINCT * FROM Stock_Price WHERE stock_label = ? AND timestamp >= "2019-04-16" COLLATE NOCASE ORDER BY timestamp);`, label, (err, rows) => {
    if (err) {
      return callback(err, null);
    } else {
      return callback(null, rows);
    }
    return callback(err, null);
  });
}


function retrieveTagTimeRecords(tag, callback){
  db.all(`SELECT timestamp FROM Time_Value WHERE tag = ? AND timestamp >= "2019-04-16" COLLATE NOCASE;`, tag, (err, rows) => {
    if (err) {
      return callback(err, null);
    } else {
      return callback(null, rows);
```

```
    }
    return callback(err, null);
  });
}

function retrieveStockTimeRecords(label, callback){
  db.all(`SELECT timestamp FROM (SELECT DISTINCT * FROM Stock_Price WHERE
stock_label = ? AND timestamp >= "2019-04-16" COLLATE NOCASE ORDER BY
timestamp);`, label, (err, rows) => {
    if (err) {
      return callback(err, null);
    } else {
      return callback(null, rows);
    }
    return callback(err, null);
  });
}

function getTagNames(callback) {
  db.all(`SELECT tag FROM Hashtag;`, (err, rows) => {
    if (err) {
      return callback(err, null);
    } else {
      return callback(null, rows);
    }
    return callback(err, null);
  });

}

/*****************************************************************************
 * DATABASE MODULE EXPORTS
 *****************************************************************************/
```

```
module.exports = {
  getTagNames,
  checkIfTagExists,
  checkIfLabelExists,
  updateTagValue,
  addNewTag,
  addTagRecord,
  addStockRecord,
  updateHastagWithStockLabel,
  updateTagConstValue,
  calculateTagDifference,
  retrieveTagValueRecords,
  retrieveStockTimeRecords,
  retrieveStockPriceRecords,
  retrieveTagTimeRecords
};
```

[redacted] [redacted]

```
{
  "name": "Enigma",
  "version": "1.1.0",
  "description": "Enigma",
  "main": "server.js",
  "scripts": {
    "test": "test"
  },
  "keywords": [
    "en1"
  ],
  "author": "[redacted]
  "license": "ISC",
  "dependencies": {
    "alphavantage": "^1.2.4",
    "create-react-app": "^2.1.8",
    "dateformat": "^3.0.3",
    "express": "^4.16.4",
    "fs": "0.0.1-security",
    "plotly": "^1.0.6",
    "sendmail": "^1.4.1",
    "sqlite3": "^4.0.0"
  }
}
FROM node:8

# DOCKERFILE CODE ADAPTED FROM THE NODEJS DOCKER DOCUMENTATION
WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install
```

[redacted] [redacted]

COPY . .

EXPOSE 8080/3001

CMD ["npm", "start"]

Appendix C: Tutorial 8 – Data Mining Module – City University (Peter Smith)

IN3011 Data Mining Week8 Tutorial

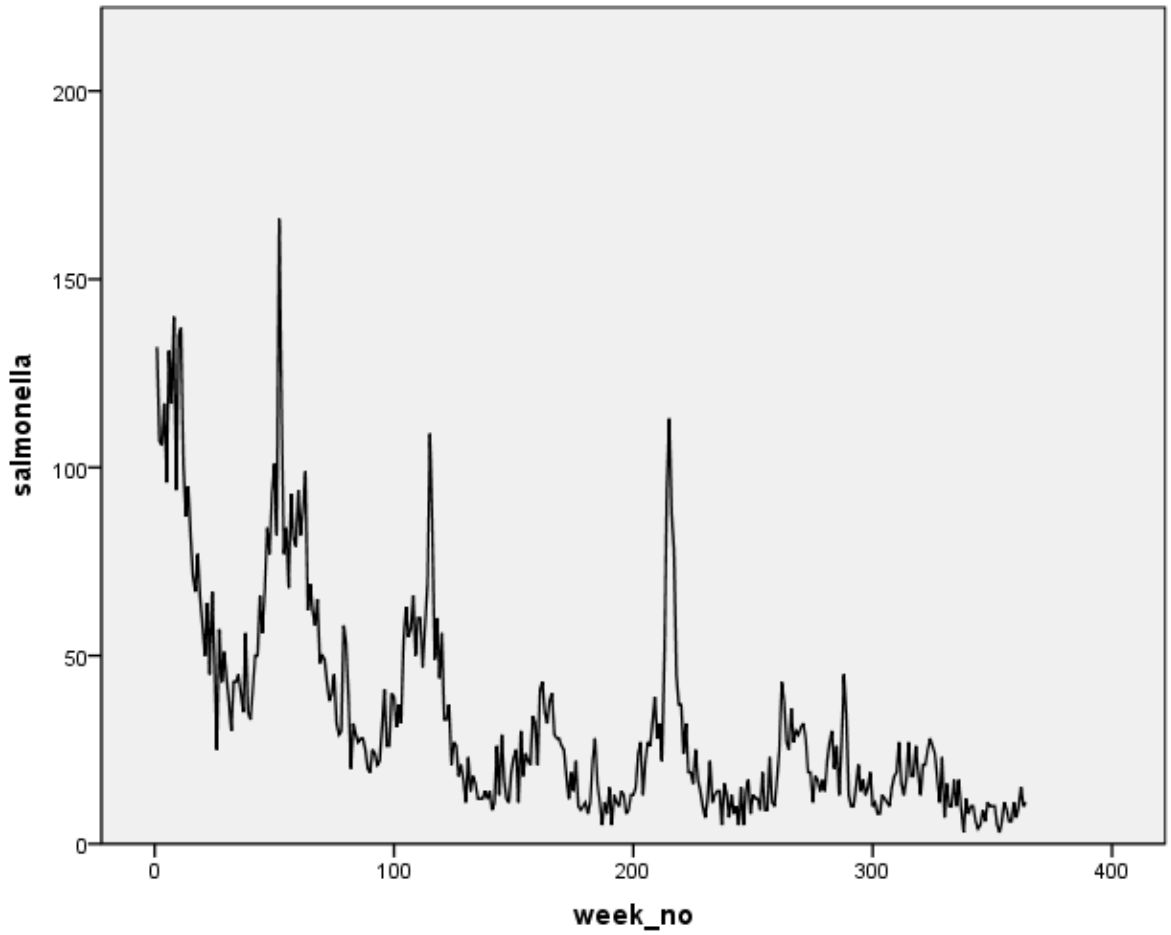
Time-Series Analysis

Displaying a Time Series

Load the datafile infections.sav which contains a time series of weekly reports from the UK of four infectious agents. Create a consistent week number by: clicking on **Transform -> Compute Variable**. Then enter week_no in target variable. Then compute week_no by **(year-1996)*52+week-26.** Then set week_no to 0 decimal places.

Click on **Graphs -> Chart-Builder**. Then drag Simple-Line to the chart display area. Drag salmonella to the y-axis and week_no to the x-axis. Then click on Element Properties and select Value. Click on OK and a time series should be displayed.

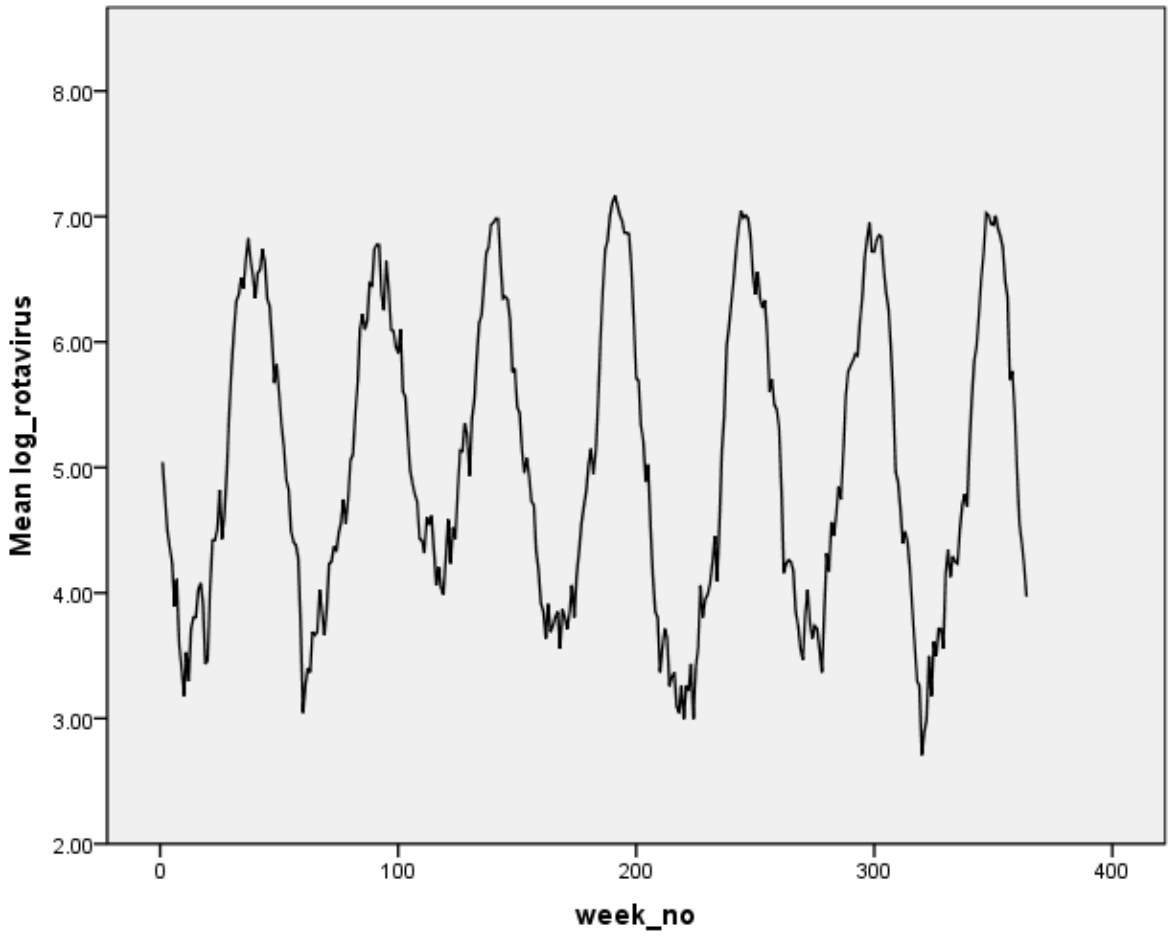
It is als



It is also possible to display time-series charts by clicking **Analyse -> Forecasting -> Sequence Charts**.

Transforming Time-Series

Sometimes it is useful to transform a time-series. Create a graph of rotavirus as above. Then click on Transform -> Compute Variable. Enter log_rotavirus in the box entitled: target variable. Now click on arithmetic function and click on Ln (natural Log) – enter rotavirus in the brackets where ? is given. Then click on OK. This should create a new variable which is the natural log of rotavirus. Now create a graph of log_rotavirus – see that the variance is a lot less than for rotavirus.



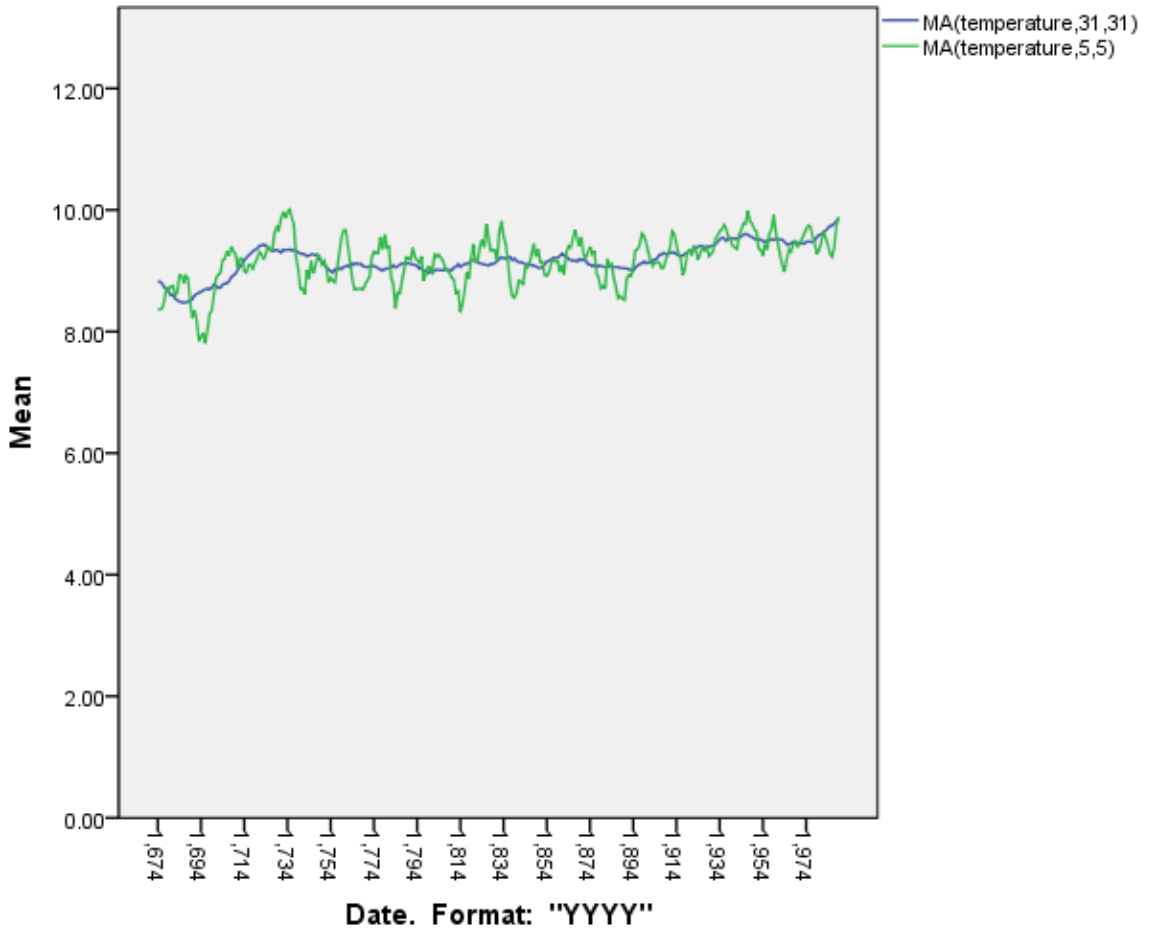
Open the datafile alltemps.sav which contains the time series of annual average temperatures recorded in Central England between 1659 and 2004.

Moving Averages

Now Click on **Transform -> Create Time Series**. Enter temperature in the variable field. Now type ma5 in the Name field in the Name and Function Area. Click on Centred Moving Average from the Function drop-down list. Enter 5 in the Span field. This specifies the order of the moving average. Click on Change and then OK. This creates a new variable in the data file.

Repeat this operation to create a moving average of order 31 (call it ma31).

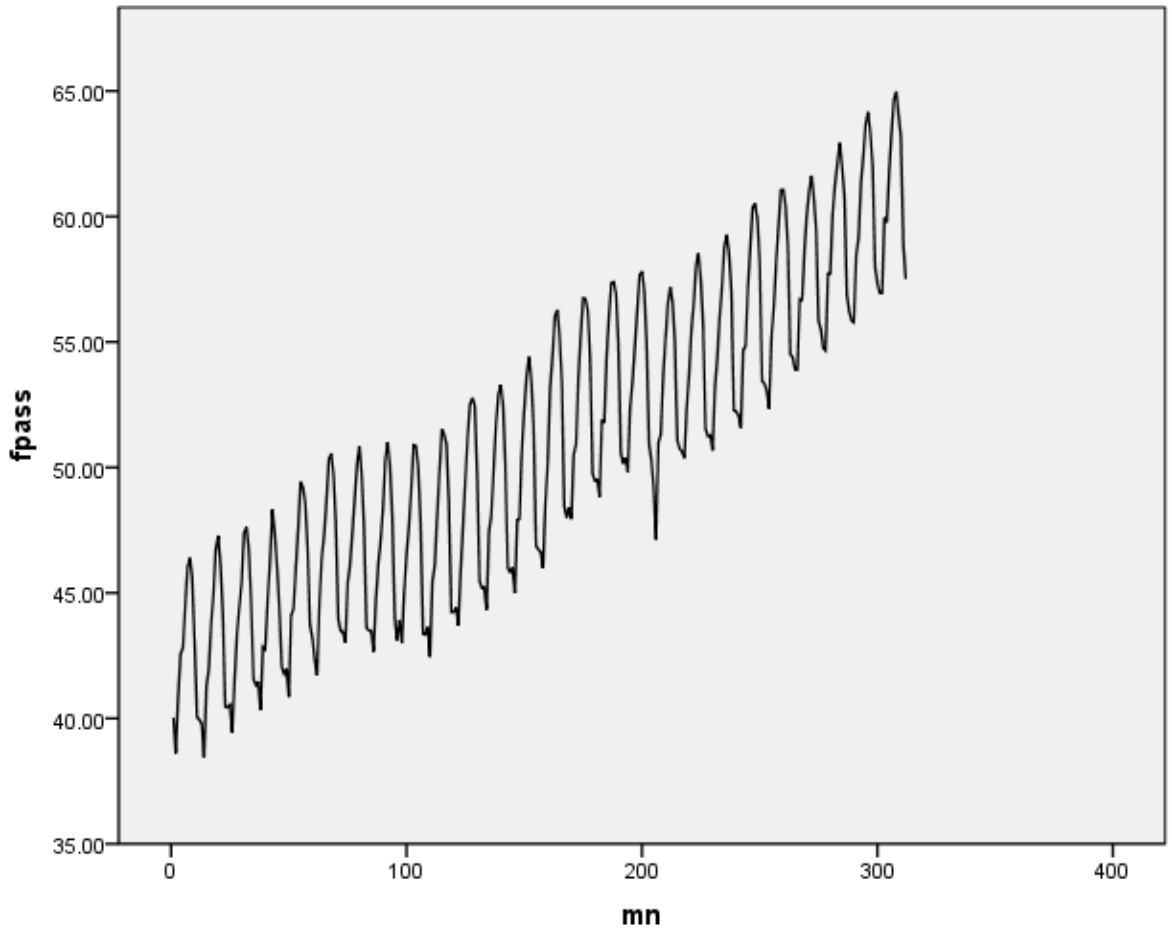
You can now display both time series using the line chart in Graphs. Remember to select Value under Element properties.



Estimating the Components of a Time Series

Open the file airline2.sav. The first thing we now want to do is numerically label each month. One cunning way of doing this is: Click on **Transform -> Compute Variable**. Create a target variable, say month_number and compute it by (Year-1974)*12+month.

When you have succeeded doing that, plot the line chart of fpass (which is the number of monthly passengers) and you should see this:



It is possible to describe time series in terms of cycles, trends and seasonality. A cycle is a regular pattern that repeats at fixed intervals. In the graph above we can see just such a pattern. But a time series can also have a trend, which is a gradual change (upwards or downwards) of the mean value or level of the time series. The time series above displays both a cycle and a trend. Seasonality is a change that follows a specific time period (e.g. monthly, annually, daily).

We may think of a time series as being built up of components: its trend, its seasonal and irregular components.

Time t and season	Trend Component	Seasonal Component	Irregular Component	Total
1 Spring	100	+20	+6.4	126.4
2 Summer	110	-35	-9.2	65.8
3 Autumn	120	-15	+4.2	109.2
4 Winter	130	+30	+12.6	172.6
5 Spring	140	+20	-5.0	155.0
6 Summer	150	-35	+5.6	120.6
7 Autumn	160	-15	-6.0	139.0
8 Winter	170	+30	+2.2	202.2

This is called an **additive decomposition model** of a time series. The trend represents an overall upwards or downwards direction, the seasonal factors describe the way the time series fluctuates seasonally, the irregular component measures variation due to other

factors. For example, if we wanted to establish global warming, we would eliminate seasonal variation, and irregular factors to reveal an underlying upwards trend. It is also possible for a time series to multiplicative in which case it will look as a though a curve rather than a line fits the trend. It is sometimes possible to transform a multiplicative model time series into an additive model time series by carrying out a log transformation.

A time series that has only a trend component and an irregular component is called a non-seasonal additive model:

$X_t = m_t + W_t$

Here m is the trend and W is the irregular component. Sometimes the trend is obscured by the irregular component (noise) one way of observing the trend is to display a moving-average.

Consider now, airline2.sav – the variable fpass clearly displays seasonality.

Click on **Analyse -> Forecasting -> Seasonal Decomposition**. Put fpass in the Variables field. Select Additive for Models. Under Moving Average Weights select Endpoints weighted by .5 Click OK.

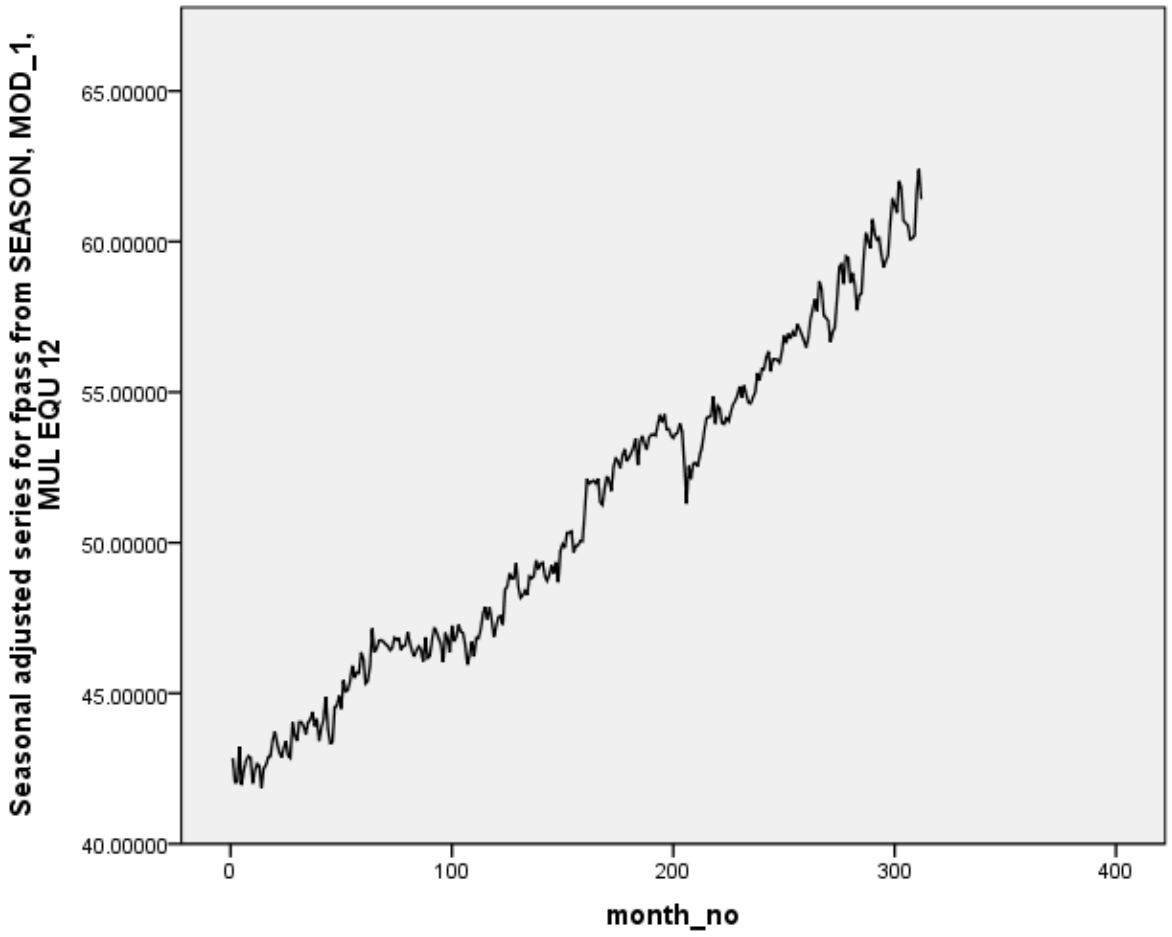
Note that is creates a table of Seasonal Factors – and 4 new variables!

Of these new variables SAS_1 is the seasonally adjusted series.

A plot of the SAS_1 is shown below:

This shows the time series with the seasonal element removed. STC_1 is a smoothed version of SAS_1 (the trend cycle). The seasonal component is in SAF_1 and ERR_1 is an estimate of the irregular component.

Now you can create a time series that contains only the irregular components as follows: Open airline4.sav (which is a tidied up version containing a variable adjusted – which is the seasonally adjusted time series). You can now create a time series called **trend** by computing the moving average (Transform -> Create Time Series) using a simple centred moving average of periodicity 11. Then you can create another variable called irregular using Transform and typing the expression adjusted – trend, then clicking OK.



Simple Exponential Smoothing

Load the file chemical.sav – you will see that it contains one variable temperature. We will start by carrying out a simple prediction using a method called exponential smoothing.

Click on **Analyse -> Forecasting -> Create Models**

Move temperature into the Dependent Variables list and then change the model to exponential smoothing. Under the Save tab – check the box for Predicted Values, then Click OK. What you will see is a new variable – this contains predicted values for time+1. Plot this time series by entering it into variables on the Sequence Chart Option (Analyse -> Forecasting -> Sequence Chart). You will see that it has been smoothed out somewhat.

How does it work?

Suppose it is a weather temperature prediction. It makes an assumption that the next value is a combination of the present and previous values based on

$\mathbf{X}_{n+1} = \mathbf{c}_0 \mathbf{x}_n + \mathbf{c}_2 \mathbf{x}_{n-1} + \mathbf{c}_2 \mathbf{x}_{n-2} + \ldots$

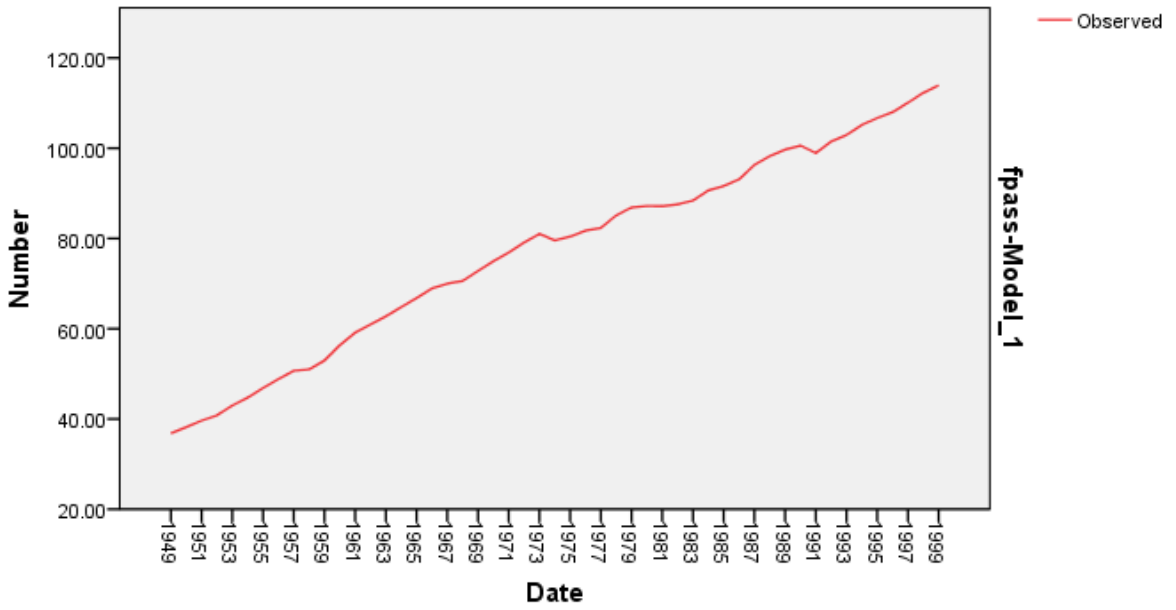
Such that $\mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 - \ldots = 1$ and $\mathbf{c}_i = \alpha\ \mathbf{(1 - \alpha)}^{\ i}$ – where α is a parameter and lower the value, the smoother the time series produced (and possibly less accurate – SPSS21 doesn't appear to allow this value to be changed – it optimises the value).

Holt's Exponential Smoothing

Now open airline5.sav. The time series fpss shows a linear increasing trend. The data shows an annual cycle (rather than one following seasons). In this case we use Holt's exponential smoothing.

Click **Analyse -> Forecasting -> Create Models**

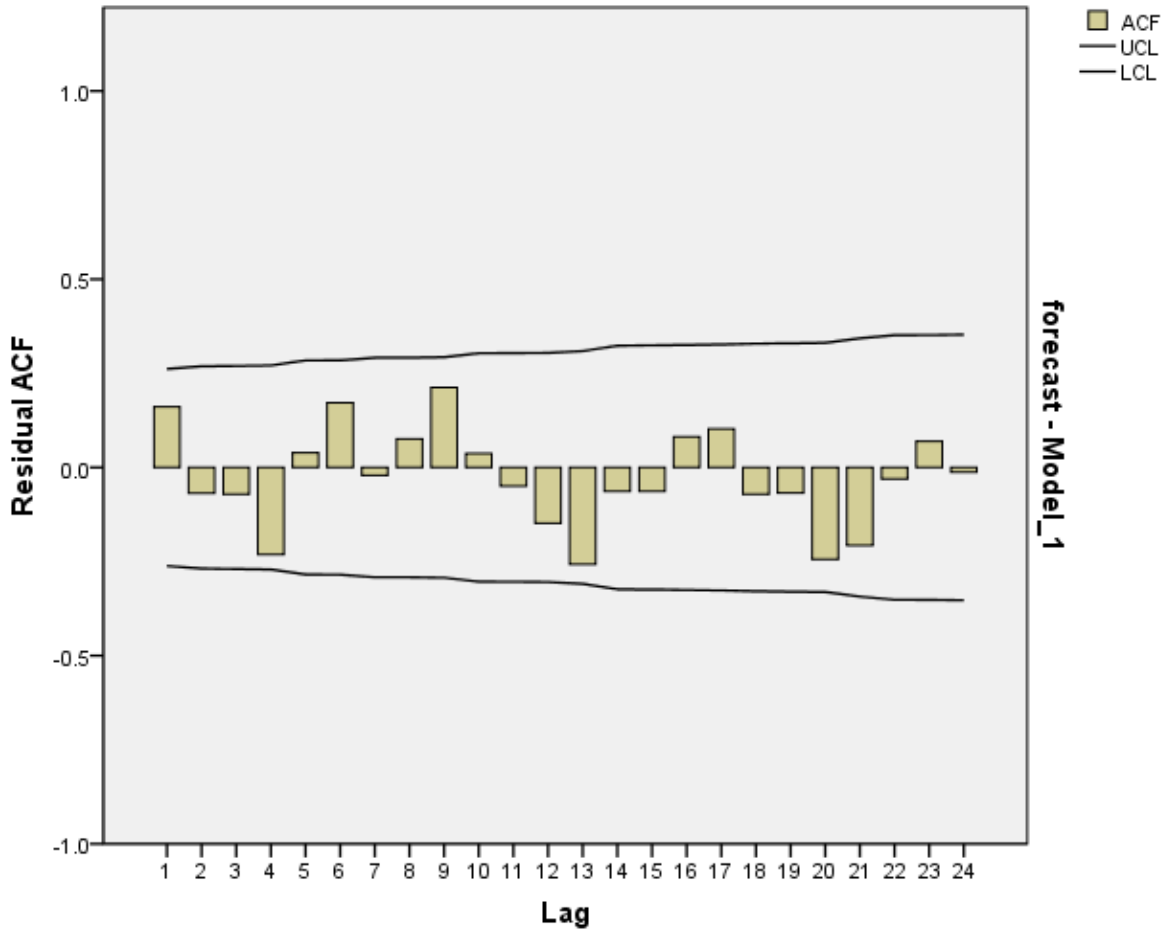
Then select Exponential Smoothing and click Criteria. Select Holt's Linear Trend. Click on the Save tab and then select Predicted Values. Then Click on OK. You should obtain:



This shows a smoothed, linearly increasing trend. The values in fpass-Model_1 give predicted values for fpass.

Correlograms and Forecasting

What is the best predictive model for a given time series? SPSS helps you here. Open airline6.sav. You will see variables fpass (original passenger volume), forecast and error (the error in the forecast). Click **Analyse -> Forecasting -> Create Models**. Now enter error in the Variables box. Use Expert Modeller as the method (this chooses the most appropriate time series model). In the Plots tab, click on Residual Autocorrelation Function (ACF)and check confidence intervals for forecasts. Click OK. One of the outputs looks like a histogram.



It is called a correlogram. The two lines drawn are significance bounds (95% significance levels). If the "bar chart" exceeds the line (each of these is called a lag) then this provides evidence against the null hypothesis which in this case is that each time series value is correlated only with the previous one. Intepreting correlograms can be difficult. Look for patterns – there is no clear pattern above. Example: the first lag means the previous value in the time series – there is a fairly strong positive correlation with the predicted value (bar above the line) – the fourth one stands out as a negative correlation (BUT – not below the significance bound.

What happens if you ask Expert modeller to deal with the variable forecast?

ARIMA Modelling

ARIMA models are integrated auto-regressive moving average models. We can only give some flavour of how they work in this tutorial.

Open the file ftse100.sav. Click **Analyse -> Forecasting -> Create Models**

Move the variable ftse into the Variables box. Select **Expert Modeler** and under criteria click on ARIMA models only. Under the Plots tab, select Residual Autocorrelation function and Partial residual autocorrelation function. (The difference between Residual Autocorrelation function and Partial residual autocorrelation function is that the ACF looks at correlations between Tn and T n-m incorporating all correlations between Tn and T n-m, whereas the PACF only considers Tn and T n-m and nothing in between). Now click OK.

You will see two correlograms. Once again there doesn't appear to be any evidence that the value for n is correlated with anything other than n-1 (so we can only predict the next value of the FTSE from the current one – within bounds of certainty. Notice under model statistics that the Ljung-Box Q test gives a significance value of 0.369 – we are looking for a value of 0.05 or less if anything other than a model based on the previous value is applicable. Notice further up it describes the model as ARIMA(0,0,0) or ARIMA(0,1,0). In general, ARIMA models can be described as ARIMA(p,d,q) where p is an auto-regressive model (a model of the form $X_{t+1} = \beta X_t + Z_t$). Generally p=0 means the "white noise" model. q is a moving average model. d is called the order of differencing (d is the number of times a time series must be differenced in order to obtain a stationary model – a model with a 0 mean). From the point of view of ftse100 prediction – an ARIMA(0,1,0) is not good news as ARIMA(0,d,0) generally means the "white noise" model.

Appendix D: Data Analysis and Generation of Output for Results

```
PREDICT THRU END.

* Time Series Modeler.

TSMODEL

  /MODELSUMMARY  PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[ RESIDACF
RESIDPACF]

  /MODELSTATISTICS  DISPLAY=YES MODELFIT=[ SRSQUARE]

  /MODELDETAILS  PRINT=[ RESIDACF RESIDPACF FORECASTS]  PLOT=[ RESIDACF
RESIDPACF]

  /SERIESPLOT OBSERVED FORECAST FIT

  /OUTPUTFILTER DISPLAY=ALLMODELS

  /AUXILIARY  CILEVEL=95 MAXACFLAGS=24

  /MISSING USERMISSING=EXCLUDE

  /MODEL DEPENDENT=enigmaValue stockPrice

    PREFIX='Model'

  /EXPERTMODELER TYPE=[ARIMA EXSMOOTH]

  /AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

[redacted] [redacted]

Notes

Output Created		17-APR-2019 08:12:34
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as missing.
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.

Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=EXCLUDE /MODEL DEPENDENT=enigmaValue stockPrice PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.47
	Elapsed Time	00:00:00.51
Use	From	First observation
	To	Last observation
Predict	From	First observation

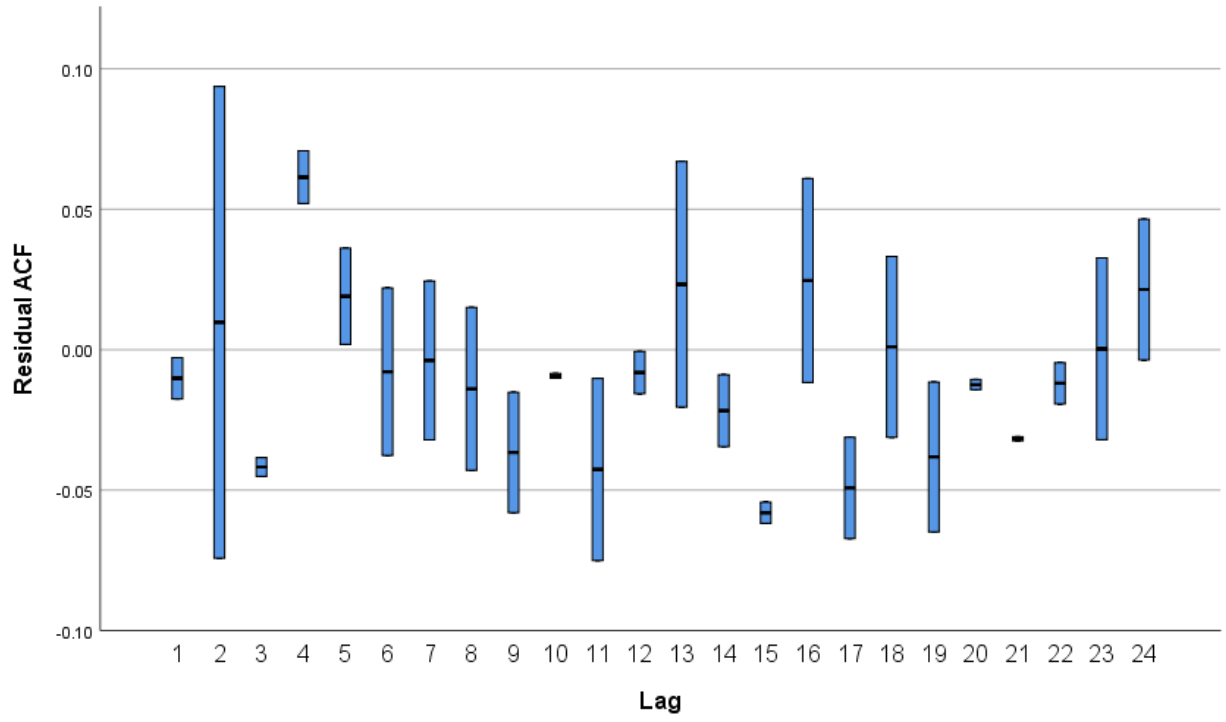
To	Last observation

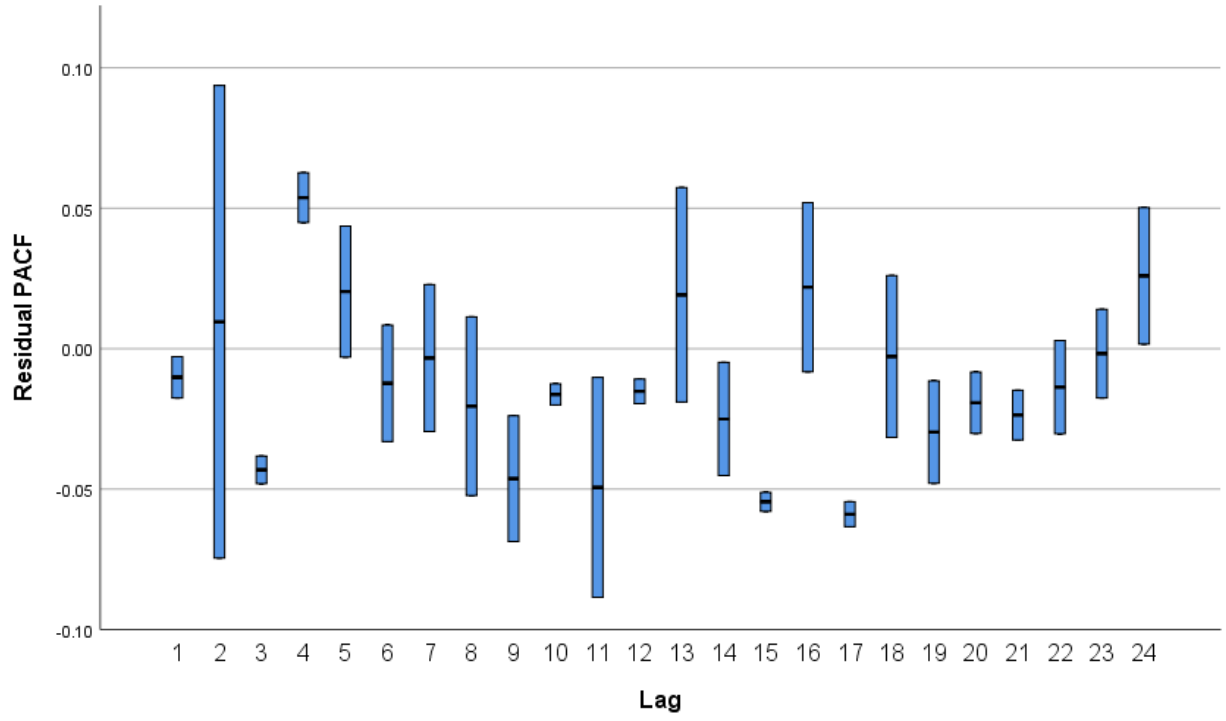
[DataSet0]

Model Description

			Model Type
Model ID	enigmaValue	Model_1	Simple
	stockPrice	Model_2	ARIMA(0,1,5)

Model Summary Chart


Residual ACF
0.10
0.05
0.00
-0.05
-0.10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Lag


Residual PACF
0.10
0.05
0.00
-0.05
-0.10
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Lag

Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10	Percentile 25
Stationary R-squared	.049	.068	.001	.097	.001	.001	.001
R-squared	.960	.028	.940	.979	.940	.940	.940
RMSE	.030	.004	.027	.033	.027	.027	.027
MAPE	11.895	16.795	.019	23.771	.019	.019	.019
MaxAPE	923.804	1305.733	.511	1847.097	.511	.511	.511
MAE	.015	.007	.010	.020	.010	.010	.010
MaxAE	.395	.187	.262	.527	.262	.262	.262
Normalized BIC	-6.983	.286	-7.185	-6.780	-7.185	-7.185	-7.185

Model Fit

Fit Statistic	Percentile 50	Percentile 75	Percentile 90	Percentile 95
Stationary R-squared	.049	.097	.097	.097
R-squared	.960	.979	.979	.979
RMSE	.030	.033	.033	.033
MAPE	11.895	23.771	23.771	23.771
MaxAPE	923.804	1847.097	1847.097	1847.097
MAE	.015	.020	.020	.020
MaxAE	.395	.527	.527	.527
Normalized BIC	-6.983	-6.780	-6.780	-6.780

[redacted] [redacted]

Residual ACF Summary

Lag	Mean	SE	Minimum	Maximum	Percentile			
					5	10	25	50
Lag 1	-.010	.010	-.018	-.003	-.018	-.018	-.018	-.010
Lag 2	.010	.119	-.074	.094	-.074	-.074	-.074	.010
Lag 3	-.042	.005	-.045	-.038	-.045	-.045	-.045	-.042
Lag 4	.061	.013	.052	.071	.052	.052	.052	.061
Lag 5	.019	.024	.002	.036	.002	.002	.002	.019
Lag 6	-.008	.042	-.038	.022	-.038	-.038	-.038	-.008
Lag 7	-.004	.040	-.032	.024	-.032	-.032	-.032	-.004
Lag 8	-.014	.041	-.043	.015	-.043	-.043	-.043	-.014
Lag 9	-.037	.030	-.058	-.015	-.058	-.058	-.058	-.037
Lag 10	-.009	.001	-.010	-.008	-.010	-.010	-.010	-.009
Lag 11	-.043	.046	-.075	-.010	-.075	-.075	-.075	-.043
Lag 12	-.008	.011	-.016	-.001	-.016	-.016	-.016	-.008
Lag 13	.023	.062	-.020	.067	-.020	-.020	-.020	.023
Lag 14	-.022	.018	-.034	-.009	-.034	-.034	-.034	-.022
Lag 15	-.058	.005	-.062	-.054	-.062	-.062	-.062	-.058
Lag 16	.025	.051	-.012	.061	-.012	-.012	-.012	.025
Lag 17	-.049	.025	-.067	-.031	-.067	-.067	-.067	-.049
Lag 18	.001	.046	-.031	.033	-.031	-.031	-.031	.001
Lag 19	-.038	.038	-.065	-.012	-.065	-.065	-.065	-.038
Lag 20	-.012	.003	-.014	-.011	-.014	-.014	-.014	-.012
Lag 21	-.032	.001	-.032	-.031	-.032	-.032	-.032	-.032
Lag 22	-.012	.010	-.019	-.005	-.019	-.019	-.019	-.012

[redacted] [redacted]

Lag 23	.000	.046	-.032	.033	-.032	-.032	-.032	.000
Lag 24	.021	.035	-.004	.046	-.004	-.004	-.004	.021

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	-.003	-.003	-.003
Lag 2	.094	.094	.094
Lag 3	-.038	-.038	-.038
Lag 4	.071	.071	.071
Lag 5	.036	.036	.036
Lag 6	.022	.022	.022
Lag 7	.024	.024	.024
Lag 8	.015	.015	.015
Lag 9	-.015	-.015	-.015
Lag 10	-.008	-.008	-.008
Lag 11	-.010	-.010	-.010
Lag 12	-.001	-.001	-.001
Lag 13	.067	.067	.067
Lag 14	-.009	-.009	-.009
Lag 15	-.054	-.054	-.054
Lag 16	.061	.061	.061
Lag 17	-.031	-.031	-.031
Lag 18	.033	.033	.033
Lag 19	-.012	-.012	-.012
Lag 20	-.011	-.011	-.011
Lag 21	-.031	-.031	-.031
Lag 22	-.005	-.005	-.005

[redacted] [redacted]

Lag 23	.033	.033	.033
Lag 24	.046	.046	.046

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	-.010	.010	-.018	-.003	-.018	-.018	-.018	-.010
Lag 2	.010	.119	-.075	.094	-.075	-.075	-.075	.010
Lag 3	-.043	.007	-.048	-.038	-.048	-.048	-.048	-.043
Lag 4	.054	.013	.045	.063	.045	.045	.045	.054
Lag 5	.020	.033	-.003	.044	-.003	-.003	-.003	.020
Lag 6	-.012	.029	-.033	.008	-.033	-.033	-.033	-.012
Lag 7	-.003	.037	-.029	.023	-.029	-.029	-.029	-.003
Lag 8	-.020	.045	-.052	.011	-.052	-.052	-.052	-.020
Lag 9	-.046	.032	-.069	-.024	-.069	-.069	-.069	-.046
Lag 10	-.016	.005	-.020	-.013	-.020	-.020	-.020	-.016
Lag 11	-.049	.055	-.089	-.010	-.089	-.089	-.089	-.049
Lag 12	-.015	.006	-.020	-.011	-.020	-.020	-.020	-.015
Lag 13	.019	.054	-.019	.057	-.019	-.019	-.019	.019
Lag 14	-.025	.028	-.045	-.005	-.045	-.045	-.045	-.025
Lag 15	-.055	.005	-.058	-.051	-.058	-.058	-.058	-.055
Lag 16	.022	.043	-.008	.052	-.008	-.008	-.008	.022
Lag 17	-.059	.006	-.063	-.055	-.063	-.063	-.063	-.059
Lag 18	-.003	.041	-.032	.026	-.032	-.032	-.032	-.003
Lag 19	-.030	.026	-.048	-.011	-.048	-.048	-.048	-.030
Lag 20	-.019	.015	-.030	-.008	-.030	-.030	-.030	-.019

[redacted] [redacted]

Lag 21	-.024	.012	-.032	-.015	-.032	-.032	-.032	-.024
Lag 22	-.014	.023	-.030	.003	-.030	-.030	-.030	-.014
Lag 23	-.002	.022	-.018	.014	-.018	-.018	-.018	-.002
Lag 24	.026	.034	.002	.050	.002	.002	.002	.026

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	-.003	-.003	-.003
Lag 2	.094	.094	.094
Lag 3	-.038	-.038	-.038
Lag 4	.063	.063	.063
Lag 5	.044	.044	.044
Lag 6	.008	.008	.008
Lag 7	.023	.023	.023
Lag 8	.011	.011	.011
Lag 9	-.024	-.024	-.024
Lag 10	-.013	-.013	-.013
Lag 11	-.010	-.010	-.010
Lag 12	-.011	-.011	-.011
Lag 13	.057	.057	.057
Lag 14	-.005	-.005	-.005
Lag 15	-.051	-.051	-.051
Lag 16	.052	.052	.052
Lag 17	-.055	-.055	-.055
Lag 18	.026	.026	.026
Lag 19	-.011	-.011	-.011
Lag 20	-.008	-.008	-.008

[redacted] [redacted]

Lag 21	-.015	-.015	-.015
Lag 22	.003	.003	.003
Lag 23	.014	.014	.014
Lag 24	.050	.050	.050

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
enigmaValue-Model_1	0	.001	21.143	17	.220
stockPrice-Model_2	0	.097	29.898	16	.019

Model Statistics

Model	Number of Outliers
enigmaValue-Model_1	0
stockPrice-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	ACF	-.003	.094	-.038	.071	.036	.022
	SE	.036	.036	.037	.037	.037	.037
stockPrice-Model_2	ACF	-.018	-.074	-.045	.052	.002	-.038
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

[redacted] [redacted]

Model		7	8	9	10	11	12
enigmaValue-Model_1	ACF	.024	.015	-.015	-.008	-.010	-.016
	SE	.037	.037	.037	.037	.037	.037
stockPrice-Model_2	ACF	-.032	-.043	-.058	-.010	-.075	-.001
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	ACF	-.020	-.009	-.054	-.012	-.067	-.031
	SE	.037	.037	.037	.037	.037	.037
stockPrice-Model_2	ACF	.067	-.034	-.062	.061	-.031	.033
	SE	.037	.038	.038	.038	.038	.038

Residual ACF

Model		19	20	21	22	23	24
enigmaValue-Model_1	ACF	-.065	-.014	-.031	-.005	-.032	-.004
	SE	.037	.038	.038	.038	.038	.038
stockPrice-Model_2	ACF	-.012	-.011	-.032	-.019	.033	.046
	SE	.038	.038	.038	.038	.038	.038

Residual PACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	PACF	-.003	.094	-.038	.063	.044	.008
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.018	-.075	-.048	.045	-.003	-.033
	SE	.037	.037	.037	.037	.037	.037

[redacted] [redacted]

Residual PACF

Model		7	8	9	10	11	12
enigmaValue-Model_1	PACF	.023	.011	-.024	-.013	-.010	-.020
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.029	-.052	-.069	-.020	-.089	-.011
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	PACF	-.019	-.005	-.051	-.008	-.055	-.032
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	.057	-.045	-.058	.052	-.063	.026
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		19	20	21	22	23	24
enigmaValue-Model_1	PACF	-.048	-.008	-.015	.003	-.018	.002
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.011	-.030	-.032	-.030	.014	.050
	SE	.037	.037	.037	.037	.037	.037

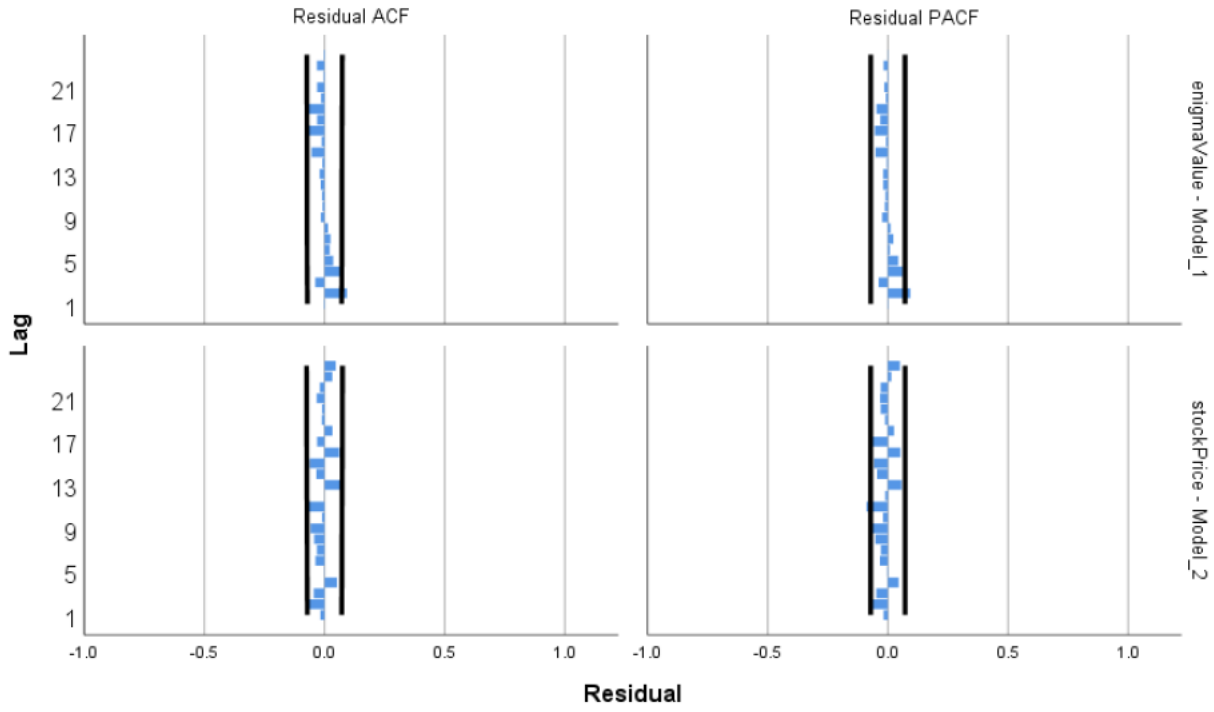
Forecast

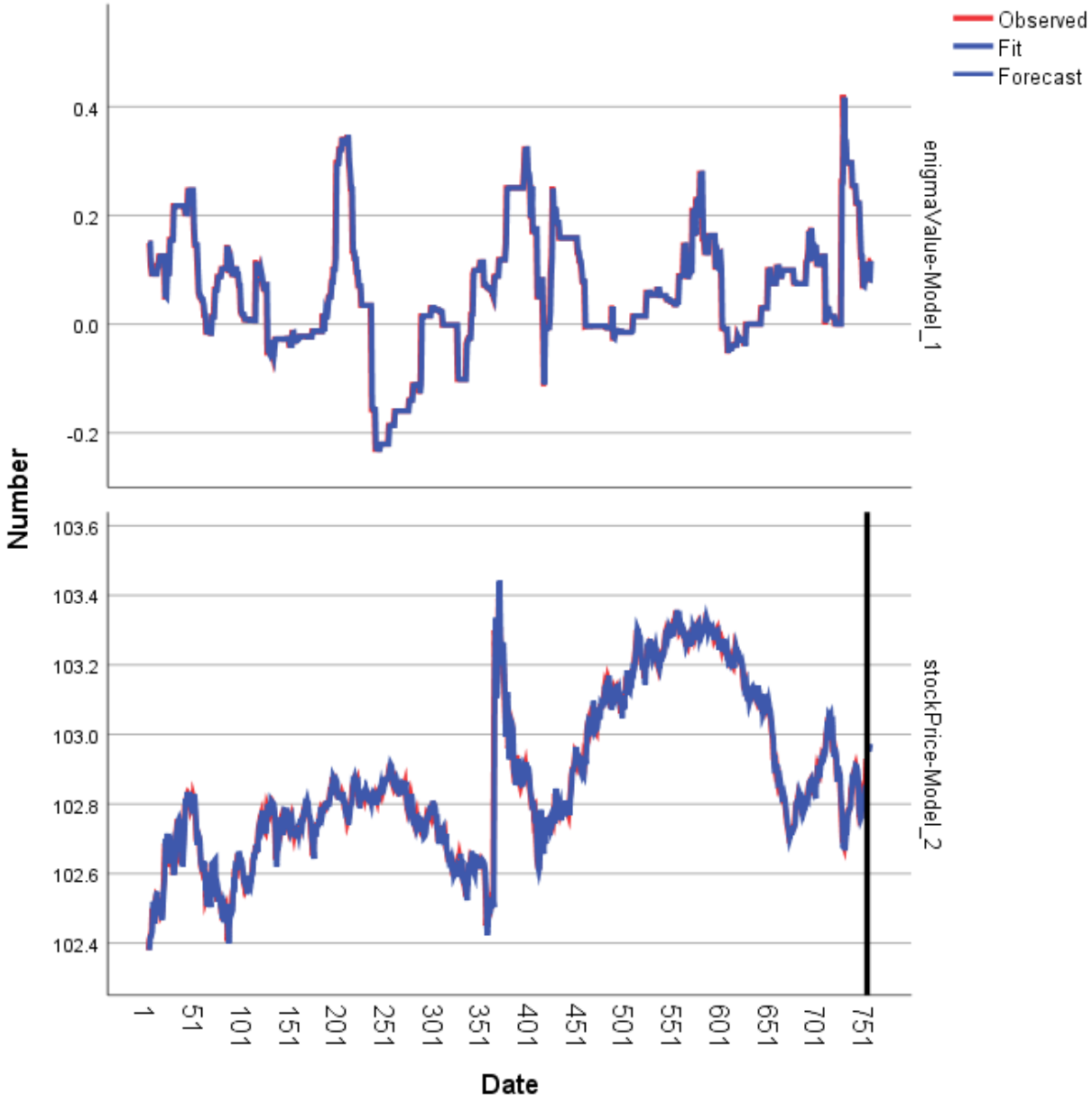
Model		752	753	754	755	756
enigmaValue-Model_1	Forecast	.	.	.	.	.
	UCL	.	.	.	.	.
	LCL	.	.	.	.	.
stockPrice-Model_2	Forecast	102.95	102.95	102.96	102.96	102.97

[redacted] [redacted]

	UCL	103.02	103.06	103.09	103.12	103.15
	LCL	102.89	102.85	102.82	102.80	102.79

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.





PREDICT THRU END.

* Time Series Modeler.

TSMODEL

/MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

/MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

/MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=stockPrice enigmaValue
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:17:06
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

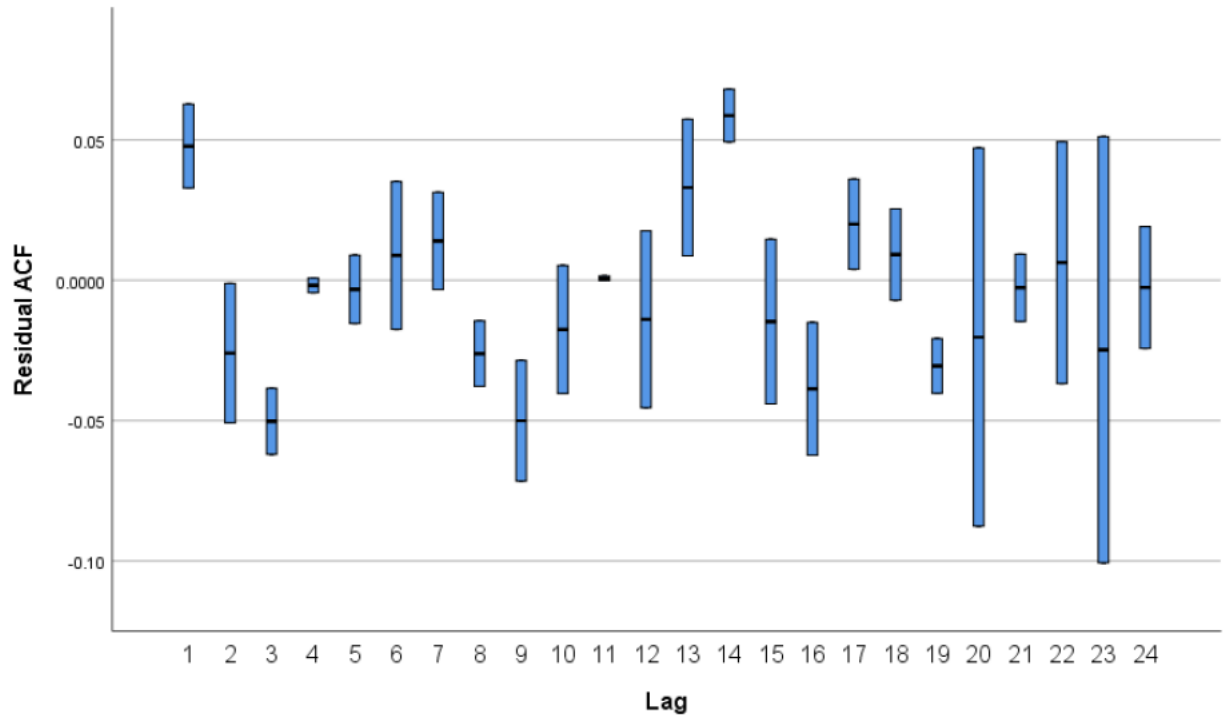
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=stockPrice enigmaValue PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.44
	Elapsed Time	00:00:00.42

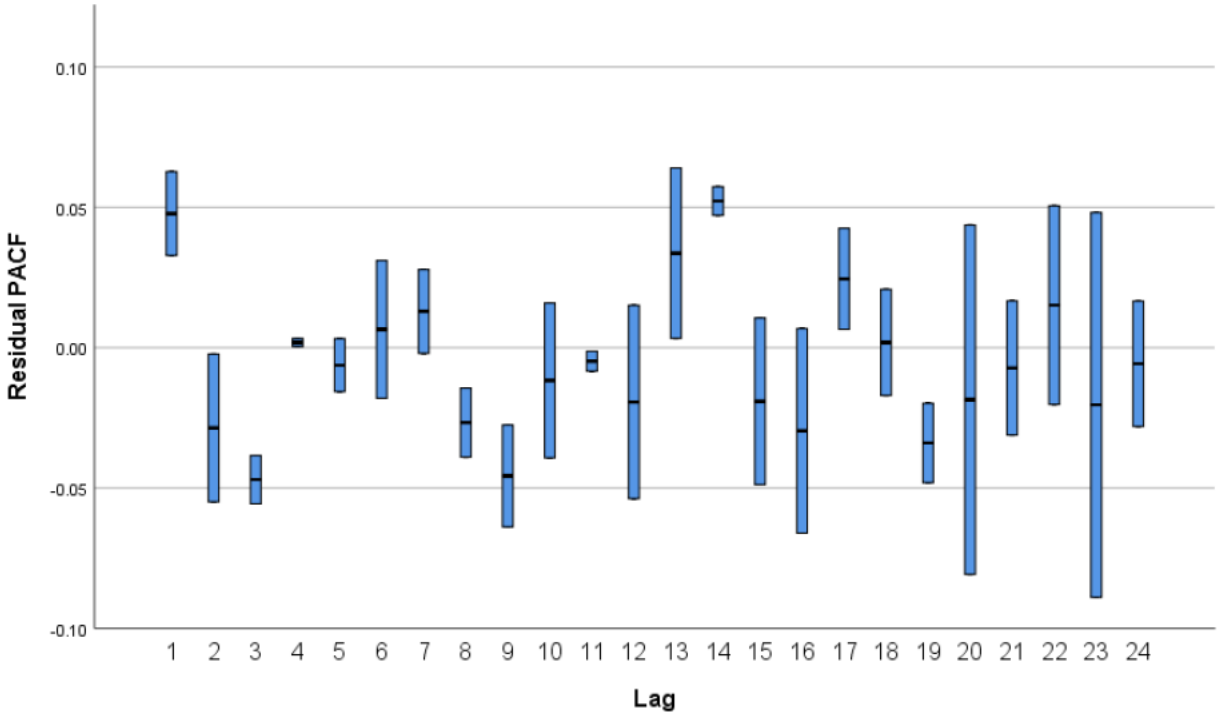
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	stockPrice	Model_1	ARIMA(0,1,5)
	enigmaValue	Model_2	ARIMA(1,0,0)

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.253	.327	.022	.484	.022	.022
R-squared	.722	.336	.484	.960	.484	.484
RMSE	.042	.008	.037	.048	.037	.037

[redacted] [redacted]

MAPE	106.732	150.915	.019	213.445	.019	.019
MaxAPE	21609.665	30560.031	.460	43218.871	.460	.460
MAE	.029	.009	.022	.036	.022	.022
MaxAE	.371	.266	.183	.560	.183	.183
Normalized BIC	-6.317	.358	-6.570	-6.064	-6.570	-6.570

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	.022	.253	.484	.484	.484
R-squared	.484	.722	.960	.960	.960
RMSE	.037	.042	.048	.048	.048
MAPE	.019	106.732	213.445	213.445	213.445
MaxAPE	.460	21609.665	43218.871	43218.871	43218.871
MAE	.022	.029	.036	.036	.036
MaxAE	.183	.371	.560	.560	.560
Normalized BIC	-6.570	-6.317	-6.064	-6.064	-6.064

Residual ACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.048	.021	.033	.063	.033	.033	.033	.048
Lag 2	-.026	.035	-.051	-.001	-.051	-.051	-.051	-.026
Lag 3	-.050	.017	-.062	-.038	-.062	-.062	-.062	-.050
Lag 4	-.002	.004	-.004	.001	-.004	-.004	-.004	-.002
Lag 5	-.003	.017	-.015	.009	-.015	-.015	-.015	-.003
Lag 6	.009	.037	-.017	.035	-.017	-.017	-.017	.009

[redacted] [redacted]

Lag 7	.014	.024	-.003	.031	-.003	-.003	-.003	.014
Lag 8	-.026	.016	-.038	-.015	-.038	-.038	-.038	-.026
Lag 9	-.050	.030	-.072	-.029	-.072	-.072	-.072	-.050
Lag 10	-.018	.032	-.040	.005	-.040	-.040	-.040	-.018
Lag 11	.001	.001	.000	.001	.000	.000	.000	.001
Lag 12	-.014	.045	-.045	.018	-.045	-.045	-.045	-.014
Lag 13	.033	.034	.009	.057	.009	.009	.009	.033
Lag 14	.059	.013	.049	.068	.049	.049	.049	.059
Lag 15	-.015	.041	-.044	.015	-.044	-.044	-.044	-.015
Lag 16	-.039	.033	-.062	-.015	-.062	-.062	-.062	-.039
Lag 17	.020	.023	.004	.036	.004	.004	.004	.020
Lag 18	.009	.023	-.007	.025	-.007	-.007	-.007	.009
Lag 19	-.031	.014	-.040	-.021	-.040	-.040	-.040	-.031
Lag 20	-.020	.095	-.088	.047	-.088	-.088	-.088	-.020
Lag 21	-.003	.017	-.015	.009	-.015	-.015	-.015	-.003
Lag 22	.006	.061	-.037	.049	-.037	-.037	-.037	.006
Lag 23	-.025	.107	-.101	.051	-.101	-.101	-.101	-.025
Lag 24	-.003	.031	-.024	.019	-.024	-.024	-.024	-.003

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.063	.063	.063
Lag 2	-.001	-.001	-.001
Lag 3	-.038	-.038	-.038
Lag 4	.001	.001	.001
Lag 5	.009	.009	.009
Lag 6	.035	.035	.035

[redacted] [redacted]

Lag 7	.031	.031	.031
Lag 8	-.015	-.015	-.015
Lag 9	-.029	-.029	-.029
Lag 10	.005	.005	.005
Lag 11	.001	.001	.001
Lag 12	.018	.018	.018
Lag 13	.057	.057	.057
Lag 14	.068	.068	.068
Lag 15	.015	.015	.015
Lag 16	-.015	-.015	-.015
Lag 17	.036	.036	.036
Lag 18	.025	.025	.025
Lag 19	-.021	-.021	-.021
Lag 20	.047	.047	.047
Lag 21	.009	.009	.009
Lag 22	.049	.049	.049
Lag 23	.051	.051	.051
Lag 24	.019	.019	.019

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.048	.021	.033	.063	.033	.033	.033	.048
Lag 2	-.029	.037	-.055	-.002	-.055	-.055	-.055	-.029
Lag 3	-.047	.012	-.056	-.038	-.056	-.056	-.056	-.047
Lag 4	.002	.002	.000	.003	.000	.000	.000	.002

[redacted] [redacted]

Lag 5	-.006	.013	-.016	.003	-.016	-.016	-.016	-.006
Lag 6	.007	.035	-.018	.031	-.018	-.018	-.018	.007
Lag 7	.013	.021	-.002	.028	-.002	-.002	-.002	.013
Lag 8	-.027	.017	-.039	-.014	-.039	-.039	-.039	-.027
Lag 9	-.046	.026	-.064	-.028	-.064	-.064	-.064	-.046
Lag 10	-.012	.039	-.039	.016	-.039	-.039	-.039	-.012
Lag 11	-.005	.005	-.008	-.001	-.008	-.008	-.008	-.005
Lag 12	-.019	.049	-.054	.015	-.054	-.054	-.054	-.019
Lag 13	.034	.043	.003	.064	.003	.003	.003	.034
Lag 14	.052	.007	.047	.057	.047	.047	.047	.052
Lag 15	-.019	.042	-.049	.011	-.049	-.049	-.049	-.019
Lag 16	-.030	.052	-.066	.007	-.066	-.066	-.066	-.030
Lag 17	.025	.025	.007	.043	.007	.007	.007	.025
Lag 18	.002	.027	-.017	.021	-.017	-.017	-.017	.002
Lag 19	-.034	.020	-.048	-.020	-.048	-.048	-.048	-.034
Lag 20	-.019	.088	-.081	.044	-.081	-.081	-.081	-.019
Lag 21	-.007	.034	-.031	.017	-.031	-.031	-.031	-.007
Lag 22	.015	.050	-.020	.051	-.020	-.020	-.020	.015
Lag 23	-.020	.097	-.089	.048	-.089	-.089	-.089	-.020
Lag 24	-.006	.032	-.028	.017	-.028	-.028	-.028	-.006

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.063	.063	.063
Lag 2	-.002	-.002	-.002
Lag 3	-.038	-.038	-.038
Lag 4	.003	.003	.003

[redacted] [redacted]

Lag 5	.003	.003	.003
Lag 6	.031	.031	.031
Lag 7	.028	.028	.028
Lag 8	-.014	-.014	-.014
Lag 9	-.028	-.028	-.028
Lag 10	.016	.016	.016
Lag 11	-.001	-.001	-.001
Lag 12	.015	.015	.015
Lag 13	.064	.064	.064
Lag 14	.057	.057	.057
Lag 15	.011	.011	.011
Lag 16	.007	.007	.007
Lag 17	.043	.043	.043
Lag 18	.021	.021	.021
Lag 19	-.020	-.020	-.020
Lag 20	.044	.044	.044
Lag 21	.017	.017	.017
Lag 22	.051	.051	.051
Lag 23	.048	.048	.048
Lag 24	.017	.017	.017

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
stockPrice-Model_1	0	.022	22.993	16	.114

[redacted] [redacted]

enigmaValue-Model_2	0	.484	12.192	17	.788

Model Statistics

Model	Number of Outliers
stockPrice-Model_1	0
enigmaValue-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
stockPrice-Model_1	ACF	.063	-.051	-.062	-.004	.009	.035
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.033	-.001	-.038	.001	-.015	-.017
	SE	.036	.036	.036	.036	.036	.036

Residual ACF

Model		7	8	9	10	11	12
stockPrice-Model_1	ACF	.031	-.015	-.072	.005	.001	-.045
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	-.003	-.038	-.029	-.040	.000	.018
	SE	.036	.036	.037	.037	.037	.037

Residual ACF

Model		13	14	15	16	17	18
stockPrice-Model_1	ACF	.057	.068	-.044	-.015	.004	-.007
	SE	.037	.037	.038	.038	.038	.038
enigmaValue-Model_2	ACF	.009	.049	.015	-.062	.036	.025

[redacted] [redacted]

	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		19	20	21	22	23	24
stockPrice-Model_1	ACF	-.021	.047	-.015	-.037	-.101	.019
	SE	.038	.038	.038	.038	.038	.038
enigmaValue-Model_2	ACF	-.040	-.088	.009	.049	.051	-.024
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		1	2	3	4	5	6
stockPrice-Model_1	PACF	.063	-.055	-.056	.000	.003	.031
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	.033	-.002	-.038	.003	-.016	-.018
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		7	8	9	10	11	12
stockPrice-Model_1	PACF	.028	-.014	-.064	.016	-.008	-.054
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	-.002	-.039	-.028	-.039	-.001	.015
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		13	14	15	16	17	18
stockPrice-Model_1	PACF	.064	.057	-.049	.007	.007	-.017
	SE	.037	.037	.037	.037	.037	.037

[redacted] [redacted]

enigmaValue-Model_2	PACF	.003	.047	.011	-.066	.043	.021
	SE	.036	.036	.036	.036	.036	.036

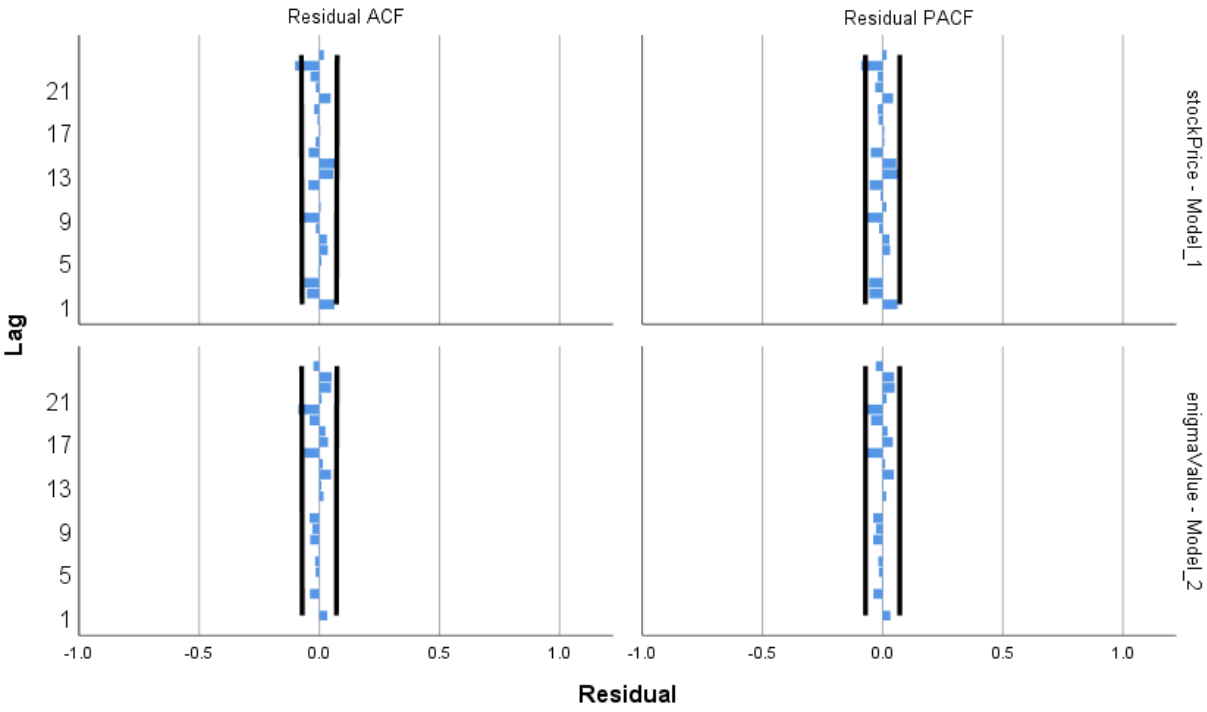
Residual PACF

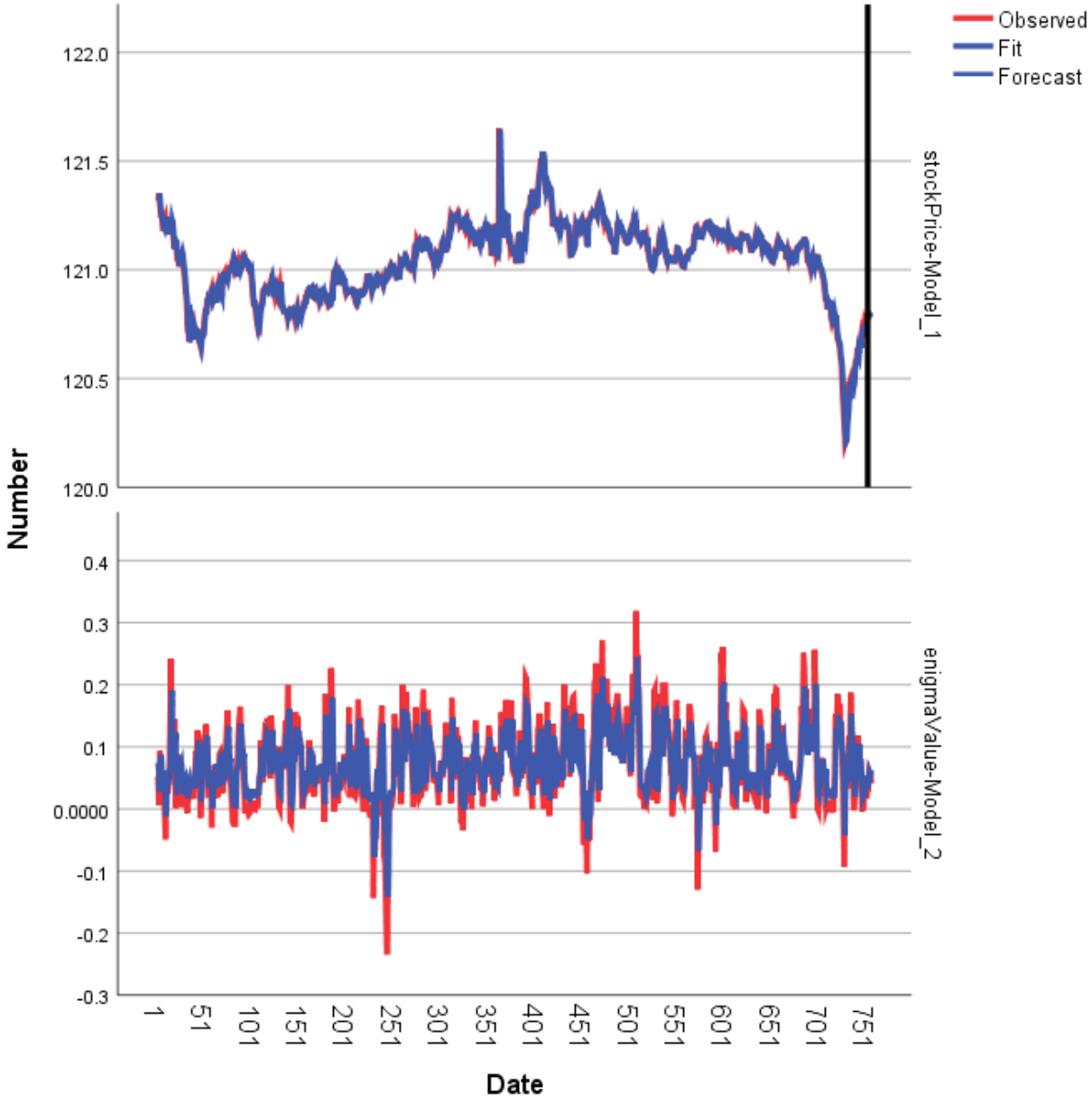
Model		19	20	21	22	23	24
stockPrice-Model_1	PACF	-.020	.044	-.031	-.020	-.089	.017
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	-.048	-.081	.017	.051	.048	-.028
	SE	.036	.036	.036	.036	.036	.036

Forecast

Model		752	753	754	755	756
stockPrice-Model_1	Forecast	120.80	120.80	120.80	120.79	120.80
	UCL	120.87	120.90	120.93	120.94	120.96
	LCL	120.72	120.70	120.67	120.65	120.64
enigmaValue-Model_2	Forecast	.	.	.	.	.
	UCL	.	.	.	.	.
	LCL	.	.	.	.	.

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.


Residual ACF
Residual PACF
stockPrice - Model_1
enigmaValue - Model_2
Lag
21
17
13
9
5
1
-1.0
-0.5
0.0
0.5
1.0
Residual



PREDICT THRU END.

* Time Series Modeler.

TSMODEL

/MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

/MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

/MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=enigmaValue stockPrice
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:18:49
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

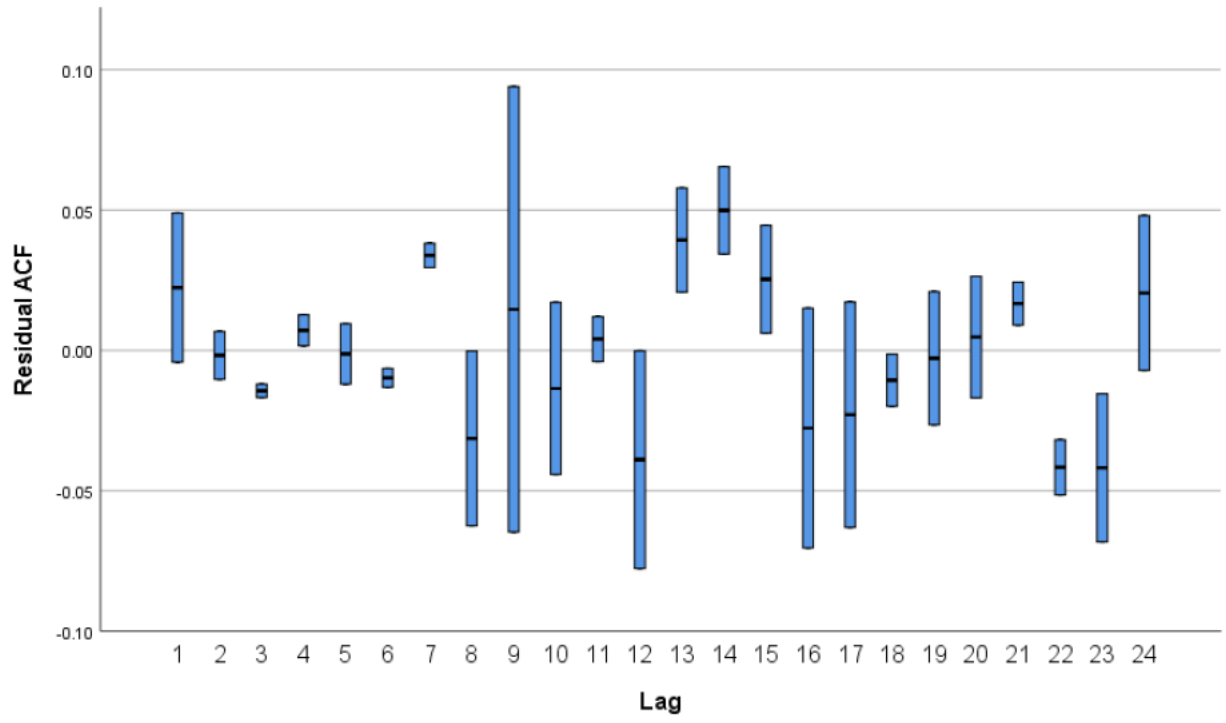
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=enigmaValue stockPrice PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.45
	Elapsed Time	00:00:00.43

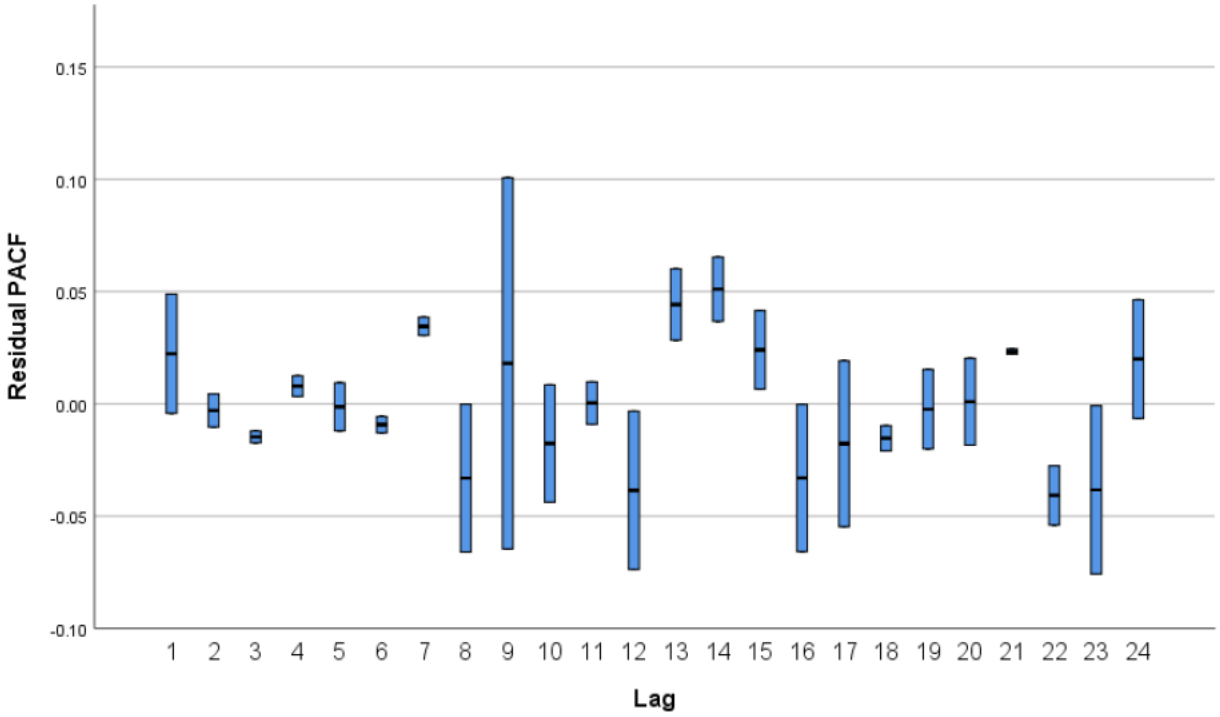
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	enigmaValue	Model_1	ARIMA(1,0,3)
	stockPrice	Model_2	Simple

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.191	.273	-.001	.384	-.001	-.001
R-squared	.689	.431	.384	.994	.384	.384
RMSE	.243	.244	.071	.416	.071	.071

[redacted] [redacted]

MAPE	95.497	135.019	.024	190.970	.024	.024
MaxAPE	10611.255	15006.202	.268	21222.242	.268	.268
MAE	.175	.174	.052	.298	.052	.052
MaxAE	1.786	2.113	.292	3.280	.292	.292
Normalized BIC	-3.505	2.490	-5.266	-1.745	-5.266	-5.266

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	-.001	.191	.384	.384	.384
R-squared	.384	.689	.994	.994	.994
RMSE	.071	.243	.416	.416	.416
MAPE	.024	95.497	190.970	190.970	190.970
MaxAPE	.268	10611.255	21222.242	21222.242	21222.242
MAE	.052	.175	.298	.298	.298
MaxAE	.292	1.786	3.280	3.280	3.280
Normalized BIC	-5.266	-3.505	-1.745	-1.745	-1.745

Residual ACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.022	.038	-.004	.049	-.004	-.004	-.004	.022
Lag 2	-.002	.012	-.010	.007	-.010	-.010	-.010	-.002
Lag 3	-.014	.003	-.017	-.012	-.017	-.017	-.017	-.014
Lag 4	.007	.008	.002	.013	.002	.002	.002	.007
Lag 5	-.001	.015	-.012	.009	-.012	-.012	-.012	-.001
Lag 6	-.010	.005	-.013	-.006	-.013	-.013	-.013	-.010

[redacted] [redacted]

Lag 7	.034	.006	.030	.038	.030	.030	.030	.034
Lag 8	-.031	.044	-.062	.000	-.062	-.062	-.062	-.031
Lag 9	.015	.112	-.065	.094	-.065	-.065	-.065	.015
Lag 10	-.014	.043	-.044	.017	-.044	-.044	-.044	-.014
Lag 11	.004	.011	-.004	.012	-.004	-.004	-.004	.004
Lag 12	-.039	.055	-.078	.000	-.078	-.078	-.078	-.039
Lag 13	.039	.026	.021	.058	.021	.021	.021	.039
Lag 14	.050	.022	.034	.065	.034	.034	.034	.050
Lag 15	.025	.027	.006	.045	.006	.006	.006	.025
Lag 16	-.028	.060	-.070	.015	-.070	-.070	-.070	-.028
Lag 17	-.023	.057	-.063	.017	-.063	-.063	-.063	-.023
Lag 18	-.011	.013	-.020	-.001	-.020	-.020	-.020	-.011
Lag 19	-.003	.034	-.027	.021	-.027	-.027	-.027	-.003
Lag 20	.005	.031	-.017	.026	-.017	-.017	-.017	.005
Lag 21	.017	.011	.009	.024	.009	.009	.009	.017
Lag 22	-.042	.014	-.052	-.032	-.052	-.052	-.052	-.042
Lag 23	-.042	.037	-.068	-.015	-.068	-.068	-.068	-.042
Lag 24	.020	.039	-.007	.048	-.007	-.007	-.007	.020

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.049	.049	.049
Lag 2	.007	.007	.007
Lag 3	-.012	-.012	-.012
Lag 4	.013	.013	.013
Lag 5	.009	.009	.009
Lag 6	-.006	-.006	-.006

[redacted] [redacted]

Lag 7	.038	.038	.038
Lag 8	.000	.000	.000
Lag 9	.094	.094	.094
Lag 10	.017	.017	.017
Lag 11	.012	.012	.012
Lag 12	.000	.000	.000
Lag 13	.058	.058	.058
Lag 14	.065	.065	.065
Lag 15	.045	.045	.045
Lag 16	.015	.015	.015
Lag 17	.017	.017	.017
Lag 18	-.001	-.001	-.001
Lag 19	.021	.021	.021
Lag 20	.026	.026	.026
Lag 21	.024	.024	.024
Lag 22	-.032	-.032	-.032
Lag 23	-.015	-.015	-.015
Lag 24	.048	.048	.048

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.022	.038	-.004	.049	-.004	-.004	-.004	.022
Lag 2	-.003	.010	-.010	.004	-.010	-.010	-.010	-.003
Lag 3	-.015	.004	-.017	-.012	-.017	-.017	-.017	-.015
Lag 4	.008	.006	.003	.012	.003	.003	.003	.008

[redacted] [redacted]

Lag 5	-.001	.015	-.012	.009	-.012	-.012	-.012	-.001
Lag 6	-.009	.005	-.013	-.006	-.013	-.013	-.013	-.009
Lag 7	.035	.006	.030	.039	.030	.030	.030	.035
Lag 8	-.033	.047	-.066	.000	-.066	-.066	-.066	-.033
Lag 9	.018	.117	-.065	.101	-.065	-.065	-.065	.018
Lag 10	-.018	.037	-.044	.008	-.044	-.044	-.044	-.018
Lag 11	.000	.013	-.009	.010	-.009	-.009	-.009	.000
Lag 12	-.039	.050	-.074	-.003	-.074	-.074	-.074	-.039
Lag 13	.044	.022	.028	.060	.028	.028	.028	.044
Lag 14	.051	.020	.037	.065	.037	.037	.037	.051
Lag 15	.024	.025	.007	.041	.007	.007	.007	.024
Lag 16	-.033	.046	-.066	.000	-.066	-.066	-.066	-.033
Lag 17	-.018	.052	-.055	.019	-.055	-.055	-.055	-.018
Lag 18	-.015	.008	-.021	-.010	-.021	-.021	-.021	-.015
Lag 19	-.002	.025	-.020	.015	-.020	-.020	-.020	-.002
Lag 20	.001	.027	-.018	.020	-.018	-.018	-.018	.001
Lag 21	.023	.002	.022	.024	.022	.022	.022	.023
Lag 22	-.041	.019	-.054	-.028	-.054	-.054	-.054	-.041
Lag 23	-.038	.053	-.076	-.001	-.076	-.076	-.076	-.038
Lag 24	.020	.037	-.006	.046	-.006	-.006	-.006	.020

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.049	.049	.049
Lag 2	.004	.004	.004
Lag 3	-.012	-.012	-.012
Lag 4	.012	.012	.012

[redacted] [redacted]

Lag 5	.009	.009	.009
Lag 6	-.006	-.006	-.006
Lag 7	.039	.039	.039
Lag 8	.000	.000	.000
Lag 9	.101	.101	.101
Lag 10	.008	.008	.008
Lag 11	.010	.010	.010
Lag 12	-.003	-.003	-.003
Lag 13	.060	.060	.060
Lag 14	.065	.065	.065
Lag 15	.041	.041	.041
Lag 16	.000	.000	.000
Lag 17	.019	.019	.019
Lag 18	-.010	-.010	-.010
Lag 19	.015	.015	.015
Lag 20	.020	.020	.020
Lag 21	.024	.024	.024
Lag 22	-.028	-.028	-.028
Lag 23	-.001	-.001	-.001
Lag 24	.046	.046	.046

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
enigmaValue-Model_1	0	.384	14.046	14	.446

[redacted] [redacted]

stockPrice-Model_2	0	-.001	24.543	17	.105

Model Statistics

Model	Number of Outliers
enigmaValue-Model_1	0
stockPrice-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	ACF	-.004	-.010	-.012	.013	.009	-.013
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	ACF	.049	.007	-.017	.002	-.012	-.006
	SE	.038	.038	.038	.038	.038	.038

Residual ACF

Model		7	8	9	10	11	12
enigmaValue-Model_1	ACF	.038	.000	-.065	-.044	.012	.000
	SE	.036	.036	.036	.037	.037	.037
stockPrice-Model_2	ACF	.030	-.062	.094	.017	-.004	-.078
	SE	.038	.038	.038	.038	.038	.038

Residual ACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	ACF	.058	.034	.006	-.070	.017	-.001
	SE	.037	.037	.037	.037	.037	.037
stockPrice-Model_2	ACF	.021	.065	.045	.015	-.063	-.020

[redacted] [redacted]

	SE	.038	.039	.039	.039	.039	.039

Residual ACF

Model		19	20	21	22	23	24
enigmaValue-Model_1	ACF	.021	-.017	.024	-.032	-.015	-.007
	SE	.037	.037	.037	.037	.037	.037
stockPrice-Model_2	ACF	-.027	.026	.009	-.052	-.068	.048
	SE	.039	.039	.039	.039	.039	.039

Residual PACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	PACF	-.004	-.010	-.012	.012	.009	-.013
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	.049	.004	-.017	.003	-.012	-.006
	SE	.038	.038	.038	.038	.038	.038

Residual PACF

Model		7	8	9	10	11	12
enigmaValue-Model_1	PACF	.039	.000	-.065	-.044	.010	-.003
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	.030	-.066	.101	.008	-.009	-.074
	SE	.038	.038	.038	.038	.038	.038

Residual PACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	PACF	.060	.037	.007	-.066	.019	-.010
	SE	.036	.036	.036	.036	.036	.036

stockPrice-Model_2	PACF	.028	.065	.041	.000	-.055	-.021
	SE	.038	.038	.038	.038	.038	.038

Residual PACF

Model		19	20	21	22	23	24
enigmaValue-Model_1	PACF	.015	-.018	.024	-.028	-.001	-.006
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.020	.020	.022	-.054	-.076	.046
	SE	.038	.038	.038	.038	.038	.038

Forecast

Model		707	708	709	710	711	712
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1228.67	1229.00	1229.26	1229.48	1229.68	1229.85
	LCL	1227.03	1226.70	1226.44	1226.22	1226.02	1225.85

Forecast

Model		713	714	715	716	717	718
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1230.01	1230.16	1230.30	1230.43	1230.56	1230.68
	LCL	1225.69	1225.54	1225.40	1225.27	1225.14	1225.02

[redacted] [redacted]

Forecast

Model		719	720	721	722	723	724
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1230.79	1230.91	1231.01	1231.12	1231.22	1231.31
	LCL	1224.91	1224.79	1224.69	1224.58	1224.48	1224.39

Forecast

Model		725	726	727	728	729	730
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1231.41	1231.50	1231.59	1231.68	1231.77	1231.85
	LCL	1224.29	1224.20	1224.11	1224.02	1223.93	1223.85

Forecast

Model		731	732	733	734	735	736
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1231.93	1232.01	1232.09	1232.17	1232.25	1232.32
	LCL	1223.77	1223.69	1223.61	1223.53	1223.45	1223.38

Forecast

[redacted] [redacted]

Model		737	738	739	740	741	742
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1232.40	1232.47	1232.54	1232.61	1232.68	1232.75
	LCL	1223.30	1223.23	1223.16	1223.09	1223.02	1222.95

Forecast

Model		743	744	745	746	747	748
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1232.82	1232.88	1232.95	1233.02	1233.08	1233.14
	LCL	1222.88	1222.82	1222.75	1222.68	1222.62	1222.56

Forecast

Model		749	750	751	752	753	754
enigmaValue-Model_1	Forecast	.	.	.	.	.	.
	UCL	.	.	.	.	.	.
	LCL	.	.	.	.	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85	1227.85	1227.85	1227.85	1227.85
	UCL	1233.21	1233.27	1233.33	1233.39	1233.45	1233.51
	LCL	1222.49	1222.43	1222.37	1222.31	1222.25	1222.19

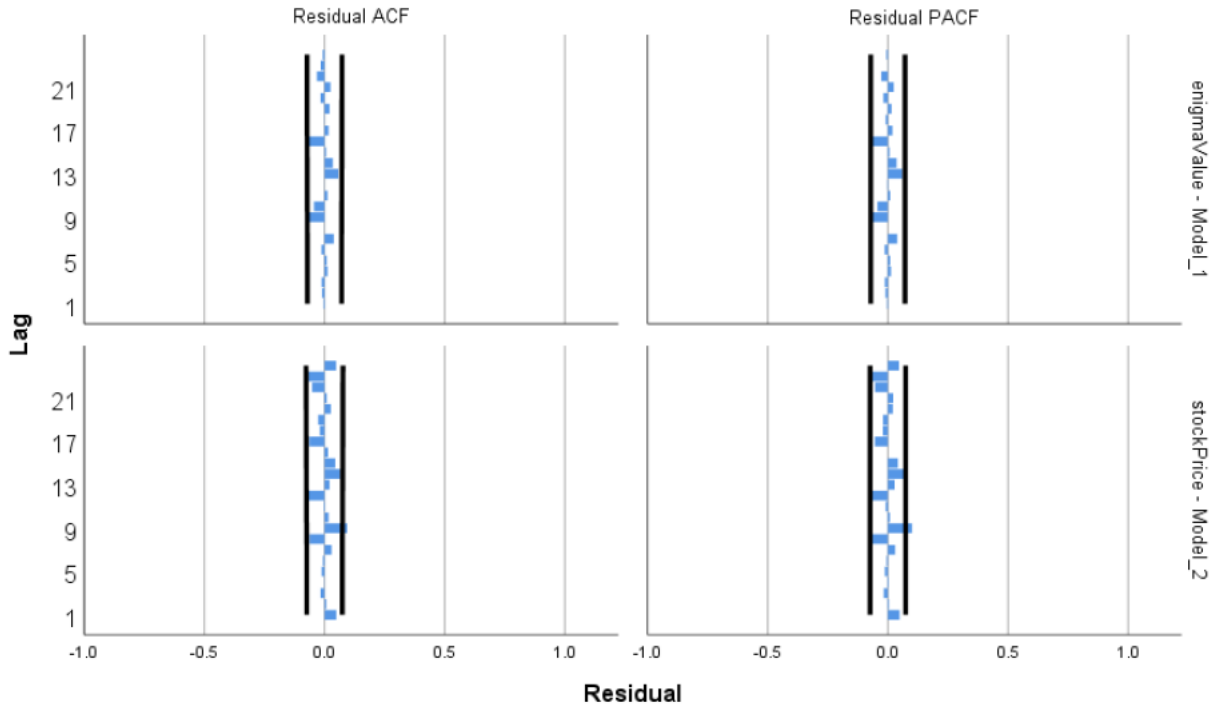
Forecast

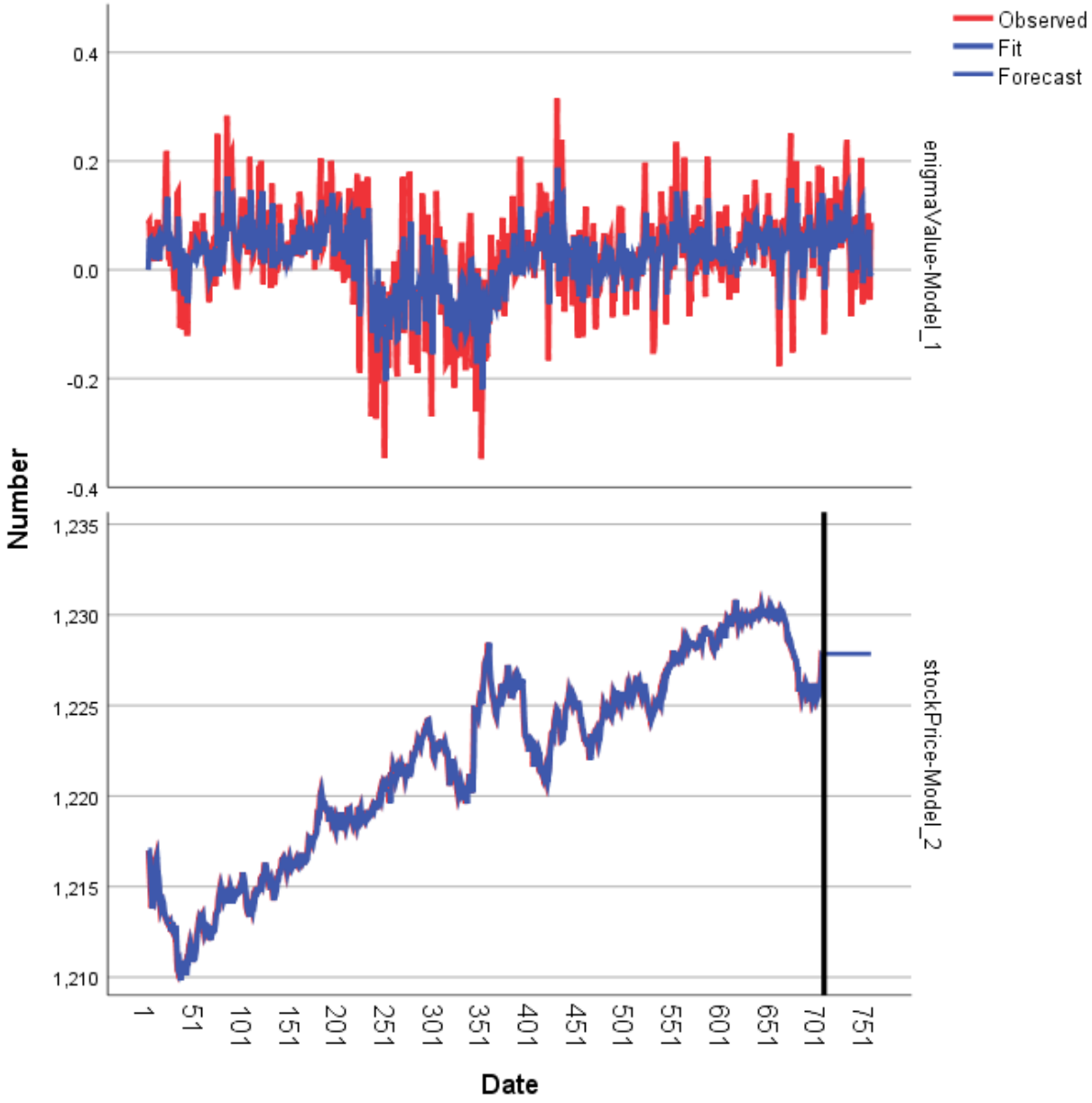
Model		755	756

[redacted] [redacted]

enigmaValue-Model_1	Forecast	.	.
	UCL	.	.
	LCL	.	.
stockPrice-Model_2	Forecast	1227.85	1227.85
	UCL	1233.57	1233.62
	LCL	1222.13	1222.08

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.





PREDICT THRU END.

* Time Series Modeler.

TSMODEL

/MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

/MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

/MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=stockPrice enigmaValue
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:20:49
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=stockPrice enigmaValue PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.45
	Elapsed Time	00:00:00.44

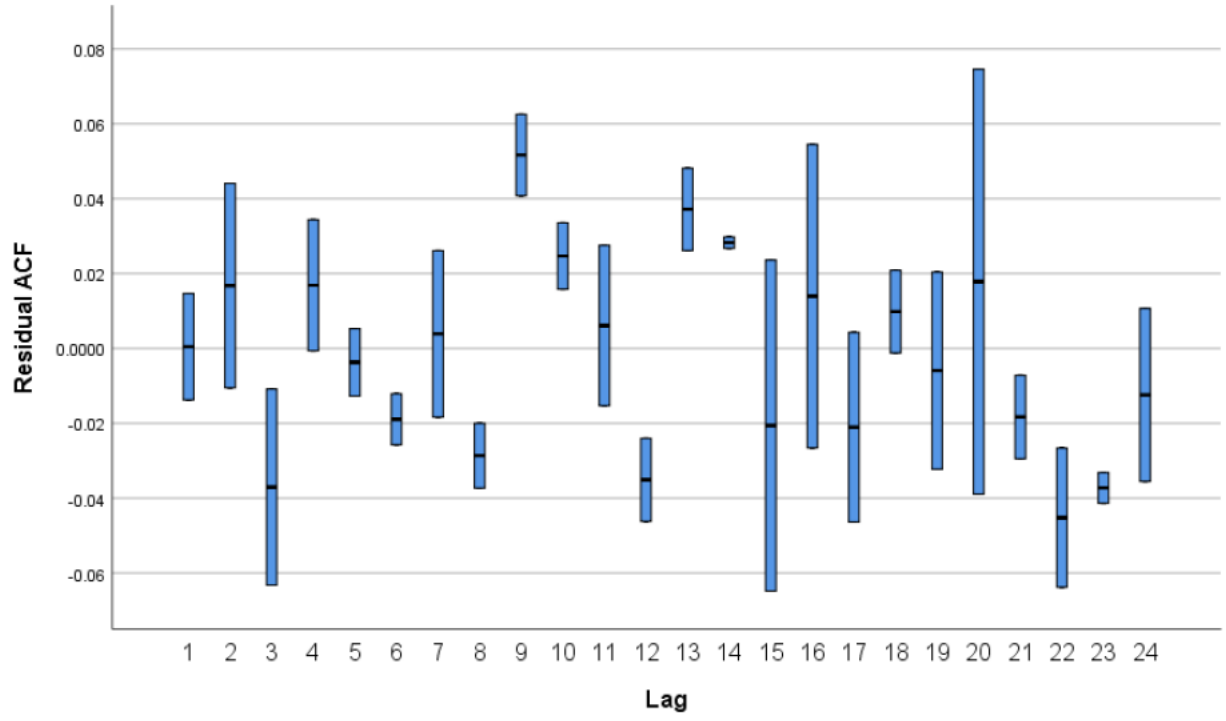
[redacted] [redacted]

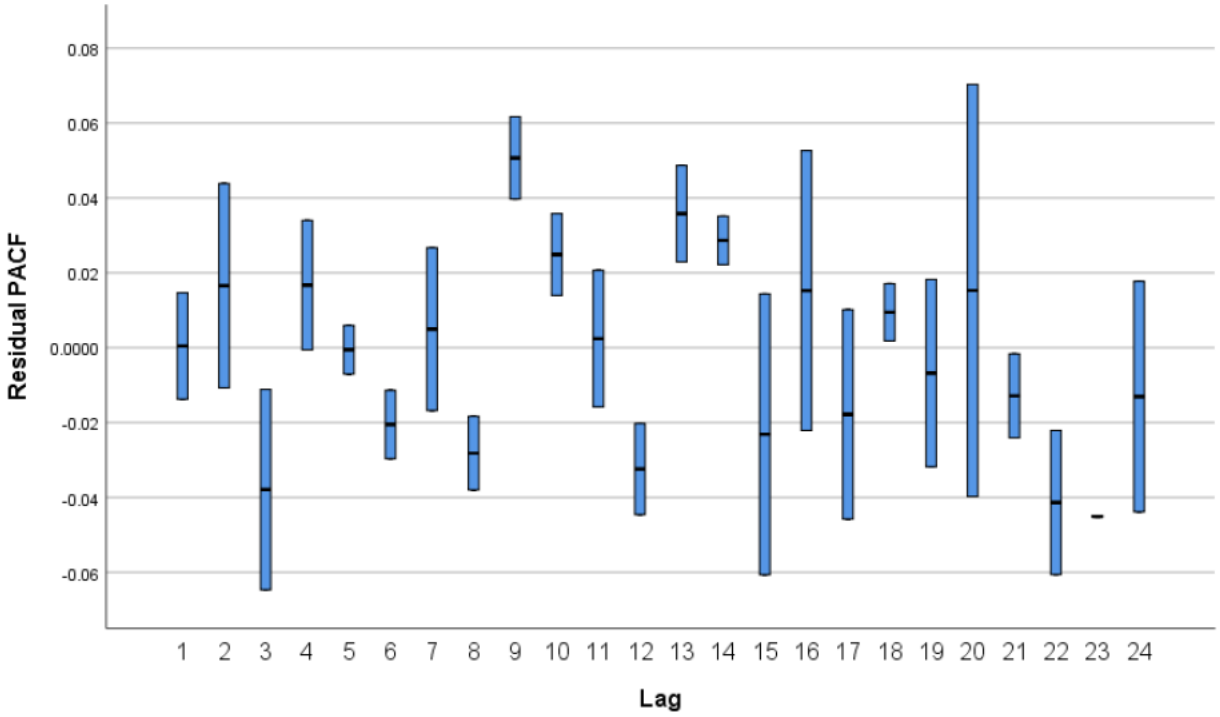
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	stockPrice	Model_1	ARIMA(0,1,1)
	enigmaValue	Model_2	ARIMA(0,0,2)

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.136	.102	.063	.208	.063	.063
R-squared	.599	.553	.208	.991	.208	.208
RMSE	.061	.003	.058	.063	.058	.058

[redacted] [redacted]

MAPE	100.599	142.239	.021	201.177	.021	.021
MaxAPE	9189.697	12995.924	.191	18379.203	.191	.191
MAE	.043	.001	.042	.044	.042	.042
MaxAE	.303	.110	.225	.381	.225	.225
Normalized BIC	-5.590	.101	-5.661	-5.519	-5.661	-5.661

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	.063	.136	.208	.208	.208
R-squared	.208	.599	.991	.991	.991
RMSE	.058	.061	.063	.063	.063
MAPE	.021	100.599	201.177	201.177	201.177
MaxAPE	.191	9189.697	18379.203	18379.203	18379.203
MAE	.042	.043	.044	.044	.044
MaxAE	.225	.303	.381	.381	.381
Normalized BIC	-5.661	-5.590	-5.519	-5.519	-5.519

Residual ACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.000	.020	-.014	.015	-.014	-.014	-.014	.000
Lag 2	.017	.039	-.011	.044	-.011	-.011	-.011	.017
Lag 3	-.037	.037	-.063	-.011	-.063	-.063	-.063	-.037
Lag 4	.017	.025	-.001	.034	-.001	-.001	-.001	.017
Lag 5	-.004	.013	-.013	.005	-.013	-.013	-.013	-.004
Lag 6	-.019	.010	-.026	-.012	-.026	-.026	-.026	-.019

[redacted] [redacted]

Lag 7	.004	.031	-.018	.026	-.018	-.018	-.018	.004
Lag 8	-.029	.012	-.037	-.020	-.037	-.037	-.037	-.029
Lag 9	.052	.015	.041	.063	.041	.041	.041	.052
Lag 10	.025	.013	.016	.034	.016	.016	.016	.025
Lag 11	.006	.030	-.015	.028	-.015	-.015	-.015	.006
Lag 12	-.035	.016	-.046	-.024	-.046	-.046	-.046	-.035
Lag 13	.037	.016	.026	.048	.026	.026	.026	.037
Lag 14	.028	.002	.027	.030	.027	.027	.027	.028
Lag 15	-.021	.063	-.065	.024	-.065	-.065	-.065	-.021
Lag 16	.014	.057	-.027	.055	-.027	-.027	-.027	.014
Lag 17	-.021	.036	-.046	.004	-.046	-.046	-.046	-.021
Lag 18	.010	.016	-.001	.021	-.001	-.001	-.001	.010
Lag 19	-.006	.037	-.032	.020	-.032	-.032	-.032	-.006
Lag 20	.018	.080	-.039	.075	-.039	-.039	-.039	.018
Lag 21	-.018	.016	-.029	-.007	-.029	-.029	-.029	-.018
Lag 22	-.045	.026	-.064	-.027	-.064	-.064	-.064	-.045
Lag 23	-.037	.006	-.041	-.033	-.041	-.041	-.041	-.037
Lag 24	-.012	.033	-.036	.011	-.036	-.036	-.036	-.012

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.015	.015	.015
Lag 2	.044	.044	.044
Lag 3	-.011	-.011	-.011
Lag 4	.034	.034	.034
Lag 5	.005	.005	.005
Lag 6	-.012	-.012	-.012

[redacted] [redacted]

Lag 7	.026	.026	.026
Lag 8	-.020	-.020	-.020
Lag 9	.063	.063	.063
Lag 10	.034	.034	.034
Lag 11	.028	.028	.028
Lag 12	-.024	-.024	-.024
Lag 13	.048	.048	.048
Lag 14	.030	.030	.030
Lag 15	.024	.024	.024
Lag 16	.055	.055	.055
Lag 17	.004	.004	.004
Lag 18	.021	.021	.021
Lag 19	.020	.020	.020
Lag 20	.075	.075	.075
Lag 21	-.007	-.007	-.007
Lag 22	-.027	-.027	-.027
Lag 23	-.033	-.033	-.033
Lag 24	.011	.011	.011

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.000	.020	-.014	.015	-.014	-.014	-.014	.000
Lag 2	.017	.039	-.011	.044	-.011	-.011	-.011	.017
Lag 3	-.038	.038	-.065	-.011	-.065	-.065	-.065	-.038
Lag 4	.017	.024	-.001	.034	-.001	-.001	-.001	.017

[redacted] [redacted]

Lag 5	-.001	.009	-.007	.006	-.007	-.007	-.007	-.001
Lag 6	-.021	.013	-.030	-.011	-.030	-.030	-.030	-.021
Lag 7	.005	.031	-.017	.027	-.017	-.017	-.017	.005
Lag 8	-.028	.014	-.038	-.018	-.038	-.038	-.038	-.028
Lag 9	.051	.016	.040	.062	.040	.040	.040	.051
Lag 10	.025	.015	.014	.036	.014	.014	.014	.025
Lag 11	.002	.026	-.016	.021	-.016	-.016	-.016	.002
Lag 12	-.032	.017	-.045	-.020	-.045	-.045	-.045	-.032
Lag 13	.036	.018	.023	.049	.023	.023	.023	.036
Lag 14	.029	.009	.022	.035	.022	.022	.022	.029
Lag 15	-.023	.053	-.061	.014	-.061	-.061	-.061	-.023
Lag 16	.015	.053	-.022	.053	-.022	-.022	-.022	.015
Lag 17	-.018	.040	-.046	.010	-.046	-.046	-.046	-.018
Lag 18	.009	.011	.002	.017	.002	.002	.002	.009
Lag 19	-.007	.035	-.032	.018	-.032	-.032	-.032	-.007
Lag 20	.015	.078	-.040	.070	-.040	-.040	-.040	.015
Lag 21	-.013	.016	-.024	-.002	-.024	-.024	-.024	-.013
Lag 22	-.041	.027	-.061	-.022	-.061	-.061	-.061	-.041
Lag 23	-.045	.000	-.045	-.045	-.045	-.045	-.045	-.045
Lag 24	-.013	.044	-.044	.018	-.044	-.044	-.044	-.013

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.015	.015	.015
Lag 2	.044	.044	.044
Lag 3	-.011	-.011	-.011
Lag 4	.034	.034	.034

[redacted] [redacted]

Lag 5	.006	.006	.006
Lag 6	-.011	-.011	-.011
Lag 7	.027	.027	.027
Lag 8	-.018	-.018	-.018
Lag 9	.062	.062	.062
Lag 10	.036	.036	.036
Lag 11	.021	.021	.021
Lag 12	-.020	-.020	-.020
Lag 13	.049	.049	.049
Lag 14	.035	.035	.035
Lag 15	.014	.014	.014
Lag 16	.053	.053	.053
Lag 17	.010	.010	.010
Lag 18	.017	.017	.017
Lag 19	.018	.018	.018
Lag 20	.070	.070	.070
Lag 21	-.002	-.002	-.002
Lag 22	-.022	-.022	-.022
Lag 23	-.045	-.045	-.045
Lag 24	.018	.018	.018

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
stockPrice-Model_1	0	.063	12.938	17	.740

[redacted] [redacted]

enigmaValue-Model_2	0	.208	16.022	16	.451

Model Statistics

Model	Number of Outliers
stockPrice-Model_1	0
enigmaValue-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
stockPrice-Model_1	ACF	.015	.044	-.063	-.001	-.013	-.026
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	-.014	-.011	-.011	.034	.005	-.012
	SE	.036	.036	.036	.036	.037	.037

Residual ACF

Model		7	8	9	10	11	12
stockPrice-Model_1	ACF	-.018	-.020	.041	.016	.028	-.046
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.026	-.037	.063	.034	-.015	-.024
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		13	14	15	16	17	18
stockPrice-Model_1	ACF	.048	.030	.024	-.027	.004	-.001
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.026	.027	-.065	.055	-.046	.021

[redacted] [redacted]

	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		19	20	21	22	23	24
stockPrice-Model_1	ACF	.020	-.039	-.029	-.027	-.041	-.036
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	-.032	.075	-.007	-.064	-.033	.011
	SE	.037	.037	.037	.037	.038	.038

Residual PACF

Model		1	2	3	4	5	6
stockPrice-Model_1	PACF	.015	.044	-.065	-.001	-.007	-.030
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	-.014	-.011	-.011	.034	.006	-.011
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		7	8	9	10	11	12
stockPrice-Model_1	PACF	-.017	-.018	.040	.014	.021	-.045
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	.027	-.038	.062	.036	-.016	-.020
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		13	14	15	16	17	18
stockPrice-Model_1	PACF	.049	.035	.014	-.022	.010	.002
	SE	.037	.037	.037	.037	.037	.037

enigmaValue-Model_2	PACF	.023	.022	-.061	.053	-.046	.017
	SE	.036	.036	.036	.036	.036	.036

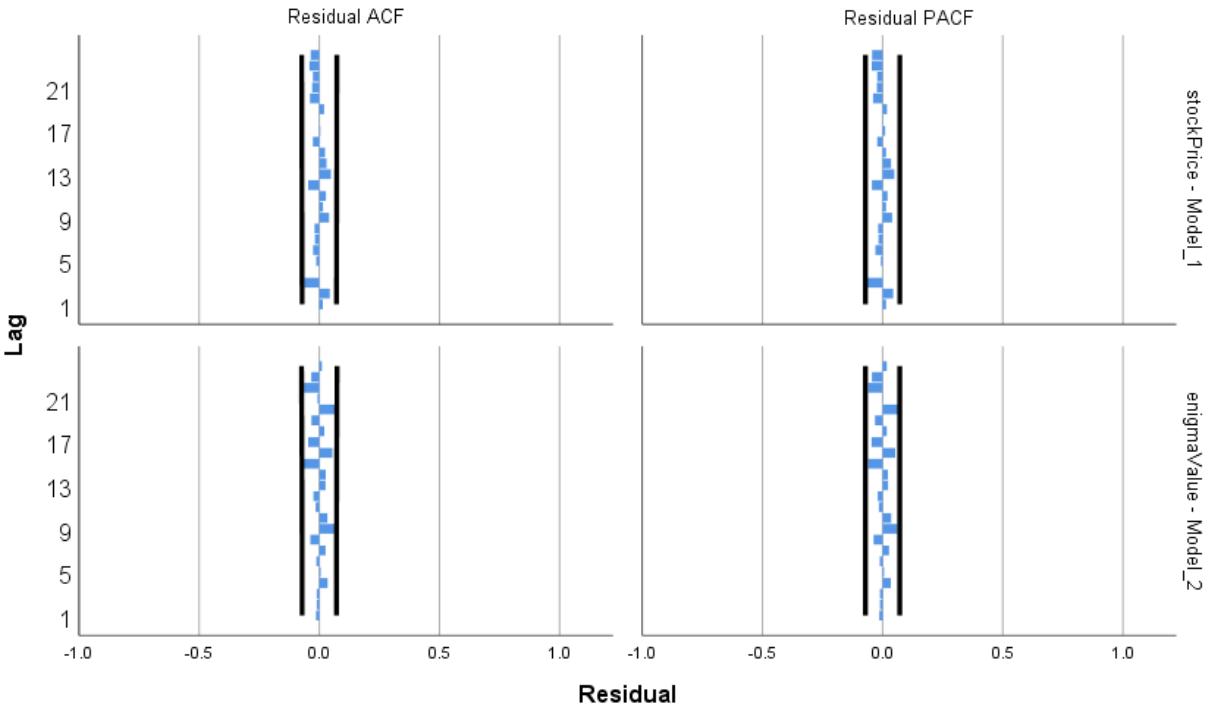
Residual PACF

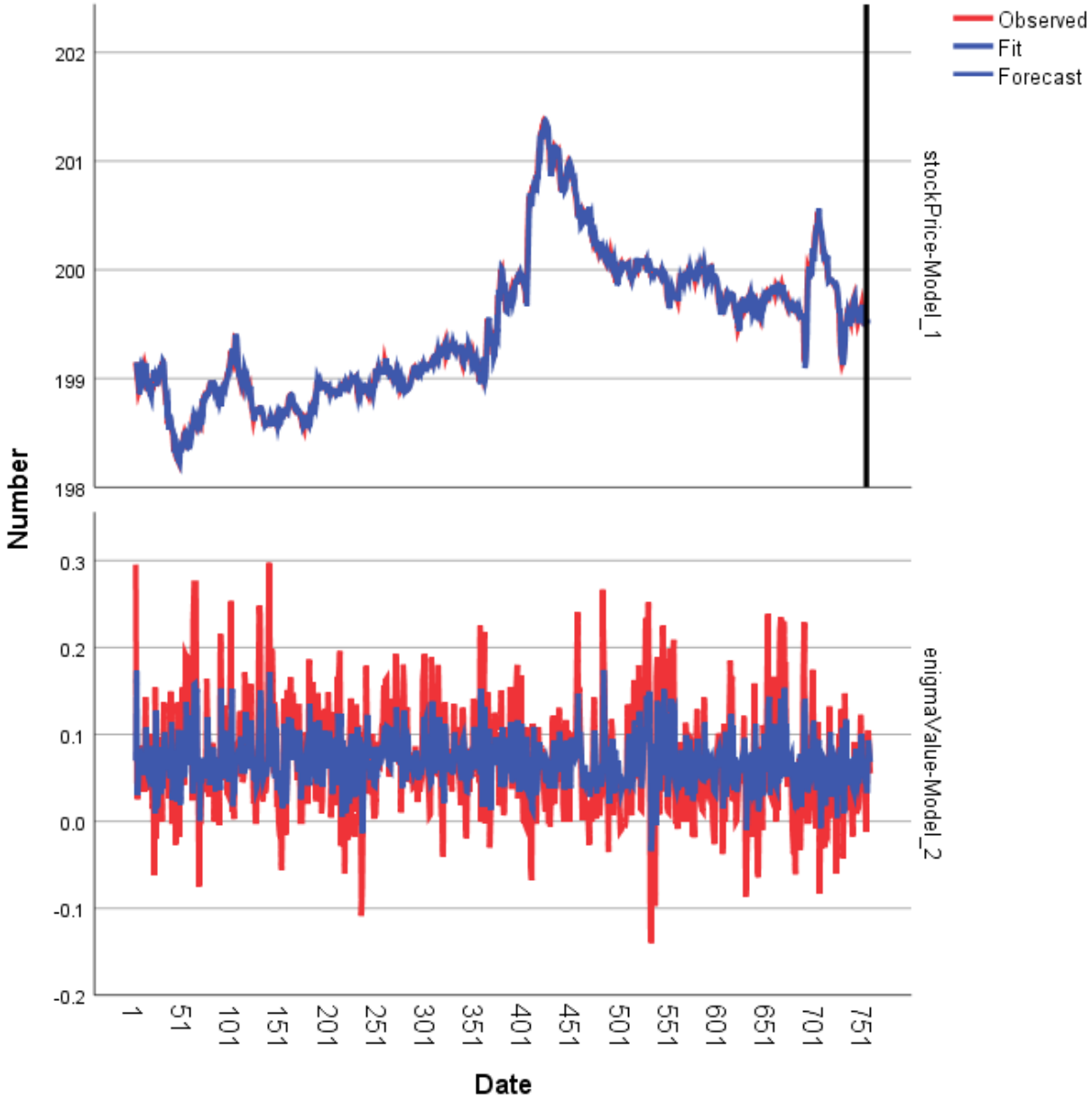
Model		19	20	21	22	23	24
stockPrice-Model_1	PACF	.018	-.040	-.024	-.022	-.045	-.044
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	-.032	.070	-.002	-.061	-.045	.018
	SE	.036	.036	.036	.036	.036	.036

Forecast

Model		752	753	754	755	756
stockPrice-Model_1	Forecast	199.52	199.52	199.52	199.52	199.52
	UCL	199.65	199.72	199.78	199.82	199.86
	LCL	199.40	199.33	199.27	199.23	199.19
enigmaValue-Model_2	Forecast	.	.	.	.	.
	UCL	.	.	.	.	.
	LCL	.	.	.	.	.

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.


Residual ACF
Residual PACF
stockPrice - Model_1
enigmaValue - Model_2
Lag
21
17
13
9
5
1
-1.0
-0.5
0.0
0.5
1.0
Residual



PREDICT THRU END.

* Time Series Modeler.

TSMODEL

  /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

  /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

  /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=stockPrice enigmaValue
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:23:17
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

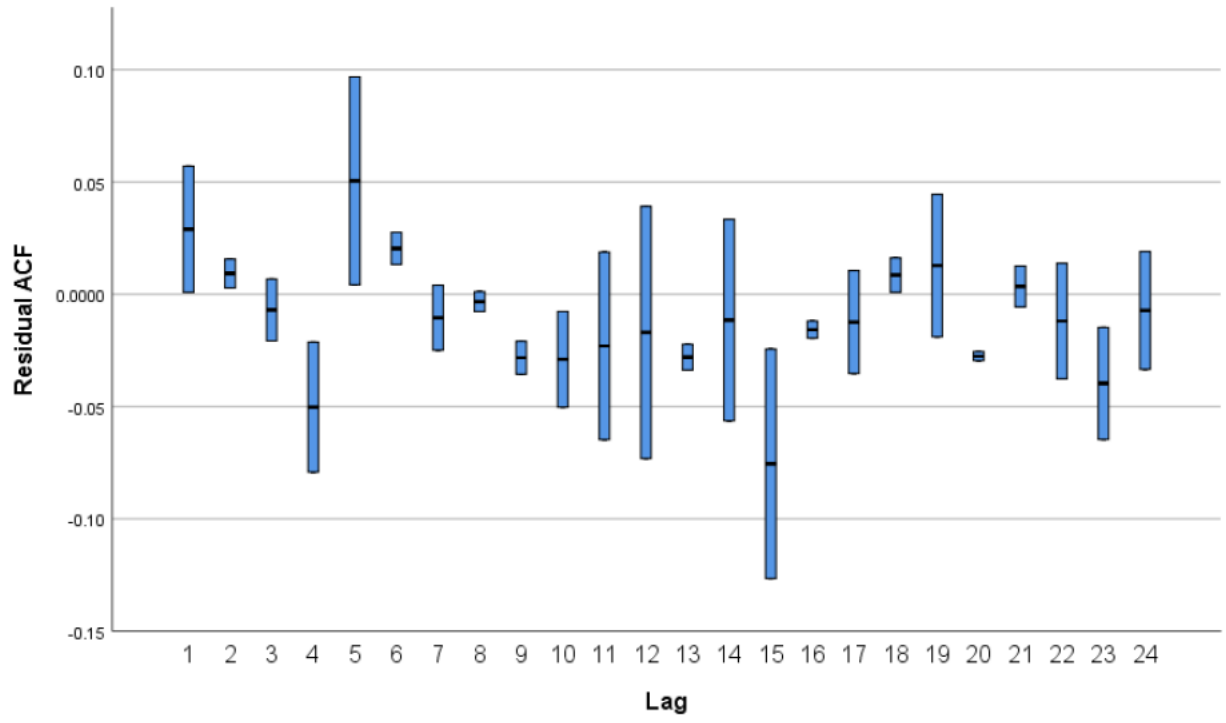
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=stockPrice enigmaValue PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.47
	Elapsed Time	00:00:00.42

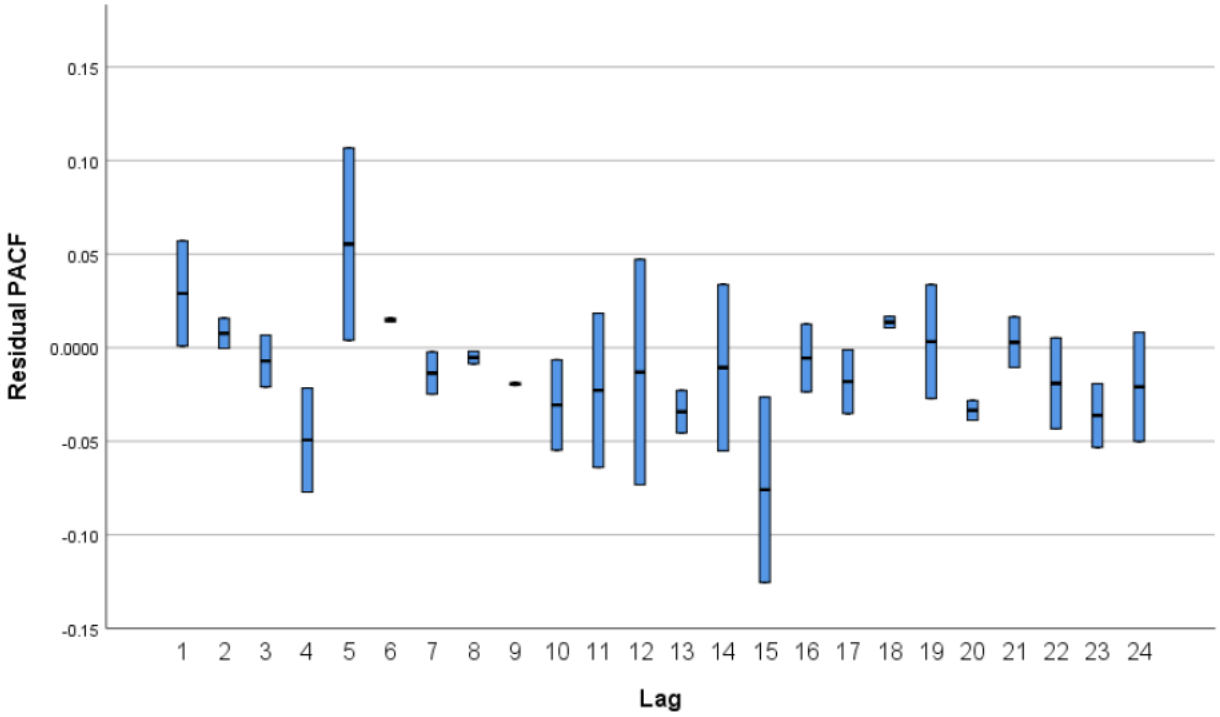
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	stockPrice	Model_1	Simple
	enigmaValue	Model_2	ARIMA(0,1,5)

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.023	.034	-.001	.047	-.001	-.001
R-squared	.897	.137	.800	.994	.800	.800
RMSE	.035	.013	.026	.044	.026	.026

[redacted] [redacted]

MAPE	24.129	34.097	.019	48.239	.019	.019
MaxAPE	2197.613	3107.404	.346	4394.880	.346	.346
MAE	.021	.010	.013	.028	.013	.013
MaxAE	.352	.209	.205	.500	.205	.205
Normalized BIC	-6.762	.744	-7.288	-6.236	-7.288	-7.288

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	-.001	.023	.047	.047	.047
R-squared	.800	.897	.994	.994	.994
RMSE	.026	.035	.044	.044	.044
MAPE	.019	24.129	48.239	48.239	48.239
MaxAPE	.346	2197.613	4394.880	4394.880	4394.880
MAE	.013	.021	.028	.028	.028
MaxAE	.205	.352	.500	.500	.500
Normalized BIC	-7.288	-6.762	-6.236	-6.236	-6.236

Residual ACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.029	.040	.001	.057	.001	.001	.001	.029
Lag 2	.009	.009	.003	.016	.003	.003	.003	.009
Lag 3	-.007	.019	-.021	.007	-.021	-.021	-.021	-.007
Lag 4	-.050	.041	-.079	-.021	-.079	-.079	-.079	-.050
Lag 5	.050	.065	.004	.097	.004	.004	.004	.050
Lag 6	.020	.010	.013	.028	.013	.013	.013	.020

[redacted] [redacted]

Lag 7	-.010	.020	-.025	.004	-.025	-.025	-.025	-.010
Lag 8	-.003	.006	-.008	.001	-.008	-.008	-.008	-.003
Lag 9	-.028	.010	-.036	-.021	-.036	-.036	-.036	-.028
Lag 10	-.029	.030	-.050	-.008	-.050	-.050	-.050	-.029
Lag 11	-.023	.059	-.065	.019	-.065	-.065	-.065	-.023
Lag 12	-.017	.079	-.073	.039	-.073	-.073	-.073	-.017
Lag 13	-.028	.008	-.034	-.022	-.034	-.034	-.034	-.028
Lag 14	-.011	.063	-.056	.033	-.056	-.056	-.056	-.011
Lag 15	-.075	.072	-.127	-.024	-.127	-.127	-.127	-.075
Lag 16	-.016	.005	-.020	-.012	-.020	-.020	-.020	-.016
Lag 17	-.012	.032	-.035	.011	-.035	-.035	-.035	-.012
Lag 18	.009	.011	.001	.016	.001	.001	.001	.009
Lag 19	.013	.045	-.019	.044	-.019	-.019	-.019	.013
Lag 20	-.028	.003	-.030	-.026	-.030	-.030	-.030	-.028
Lag 21	.003	.013	-.006	.013	-.006	-.006	-.006	.003
Lag 22	-.012	.036	-.038	.014	-.038	-.038	-.038	-.012
Lag 23	-.040	.035	-.065	-.015	-.065	-.065	-.065	-.040
Lag 24	-.007	.037	-.033	.019	-.033	-.033	-.033	-.007

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.057	.057	.057
Lag 2	.016	.016	.016
Lag 3	.007	.007	.007
Lag 4	-.021	-.021	-.021
Lag 5	.097	.097	.097
Lag 6	.028	.028	.028

[redacted] [redacted]

Lag 7	.004	.004	.004
Lag 8	.001	.001	.001
Lag 9	-.021	-.021	-.021
Lag 10	-.008	-.008	-.008
Lag 11	.019	.019	.019
Lag 12	.039	.039	.039
Lag 13	-.022	-.022	-.022
Lag 14	.033	.033	.033
Lag 15	-.024	-.024	-.024
Lag 16	-.012	-.012	-.012
Lag 17	.011	.011	.011
Lag 18	.016	.016	.016
Lag 19	.044	.044	.044
Lag 20	-.026	-.026	-.026
Lag 21	.013	.013	.013
Lag 22	.014	.014	.014
Lag 23	-.015	-.015	-.015
Lag 24	.019	.019	.019

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.029	.040	.001	.057	.001	.001	.001	.029
Lag 2	.008	.011	.000	.016	.000	.000	.000	.008
Lag 3	-.007	.020	-.021	.007	-.021	-.021	-.021	-.007
Lag 4	-.049	.039	-.077	-.022	-.077	-.077	-.077	-.049

[redacted] [redacted]

Lag 5	.055	.073	.004	.107	.004	.004	.004	.055
Lag 6	.015	.001	.014	.016	.014	.014	.014	.015
Lag 7	-.014	.016	-.025	-.002	-.025	-.025	-.025	-.014
Lag 8	-.005	.005	-.009	-.002	-.009	-.009	-.009	-.005
Lag 9	-.019	.001	-.020	-.019	-.020	-.020	-.020	-.019
Lag 10	-.031	.034	-.055	-.007	-.055	-.055	-.055	-.031
Lag 11	-.023	.058	-.064	.018	-.064	-.064	-.064	-.023
Lag 12	-.013	.085	-.073	.047	-.073	-.073	-.073	-.013
Lag 13	-.034	.016	-.046	-.023	-.046	-.046	-.046	-.034
Lag 14	-.011	.063	-.055	.034	-.055	-.055	-.055	-.011
Lag 15	-.076	.070	-.125	-.026	-.125	-.125	-.125	-.076
Lag 16	-.006	.026	-.024	.013	-.024	-.024	-.024	-.006
Lag 17	-.018	.024	-.035	-.001	-.035	-.035	-.035	-.018
Lag 18	.014	.004	.011	.017	.011	.011	.011	.014
Lag 19	.003	.043	-.027	.034	-.027	-.027	-.027	.003
Lag 20	-.034	.007	-.039	-.028	-.039	-.039	-.039	-.034
Lag 21	.003	.019	-.011	.016	-.011	-.011	-.011	.003
Lag 22	-.019	.034	-.043	.005	-.043	-.043	-.043	-.019
Lag 23	-.036	.024	-.053	-.019	-.053	-.053	-.053	-.036
Lag 24	-.021	.041	-.050	.008	-.050	-.050	-.050	-.021

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.057	.057	.057
Lag 2	.016	.016	.016
Lag 3	.007	.007	.007
Lag 4	-.022	-.022	-.022

[redacted] [redacted]

Lag 5	.107	.107	.107
Lag 6	.016	.016	.016
Lag 7	-.002	-.002	-.002
Lag 8	-.002	-.002	-.002
Lag 9	-.019	-.019	-.019
Lag 10	-.007	-.007	-.007
Lag 11	.018	.018	.018
Lag 12	.047	.047	.047
Lag 13	-.023	-.023	-.023
Lag 14	.034	.034	.034
Lag 15	-.026	-.026	-.026
Lag 16	.013	.013	.013
Lag 17	-.001	-.001	-.001
Lag 18	.017	.017	.017
Lag 19	.034	.034	.034
Lag 20	-.028	-.028	-.028
Lag 21	.016	.016	.016
Lag 22	.005	.005	.005
Lag 23	-.019	-.019	-.019
Lag 24	.008	.008	.008

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
stockPrice-Model_1	0	-.001	24.867	17	.098

[redacted] [redacted]

enigmaValue-Model_2	0	.047	22.662	15	.092

Model Statistics

Model	Number of Outliers
stockPrice-Model_1	0
enigmaValue-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
stockPrice-Model_1	ACF	.057	.003	-.021	-.079	.097	.028
	SE	.036	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.001	.016	.007	-.021	.004	.013
	SE	.036	.036	.036	.036	.036	.036

Residual ACF

Model		7	8	9	10	11	12
stockPrice-Model_1	ACF	.004	.001	-.036	-.050	-.065	.039
	SE	.037	.037	.037	.037	.037	.038
enigmaValue-Model_2	ACF	-.025	-.008	-.021	-.008	.019	-.073
	SE	.036	.036	.036	.036	.036	.036

Residual ACF

Model		13	14	15	16	17	18
stockPrice-Model_1	ACF	-.034	.033	-.024	-.012	.011	.001
	SE	.038	.038	.038	.038	.038	.038
enigmaValue-Model_2	ACF	-.022	-.056	-.127	-.020	-.035	.016

[redacted] [redacted]

	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		19	20	21	22	23	24
stockPrice-Model_1	ACF	.044	-.030	.013	-.038	-.065	.019
	SE	.038	.038	.038	.038	.038	.038
enigmaValue-Model_2	ACF	-.019	-.026	-.006	.014	-.015	-.033
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		1	2	3	4	5	6
stockPrice-Model_1	PACF	.057	.000	-.021	-.077	.107	.016
	SE	.036	.036	.036	.036	.036	.036
enigmaValue-Model_2	PACF	.001	.016	.007	-.022	.004	.014
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		7	8	9	10	11	12
stockPrice-Model_1	PACF	-.002	-.002	-.019	-.055	-.064	.047
	SE	.036	.036	.036	.036	.036	.036
enigmaValue-Model_2	PACF	-.025	-.009	-.020	-.007	.018	-.073
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		13	14	15	16	17	18
stockPrice-Model_1	PACF	-.046	.034	-.026	.013	-.001	.011
	SE	.036	.036	.036	.036	.036	.036

enigmaValue-Model_2	PACF	-.023	-.055	-.125	-.024	-.035	.017
	SE	.036	.036	.036	.036	.036	.036

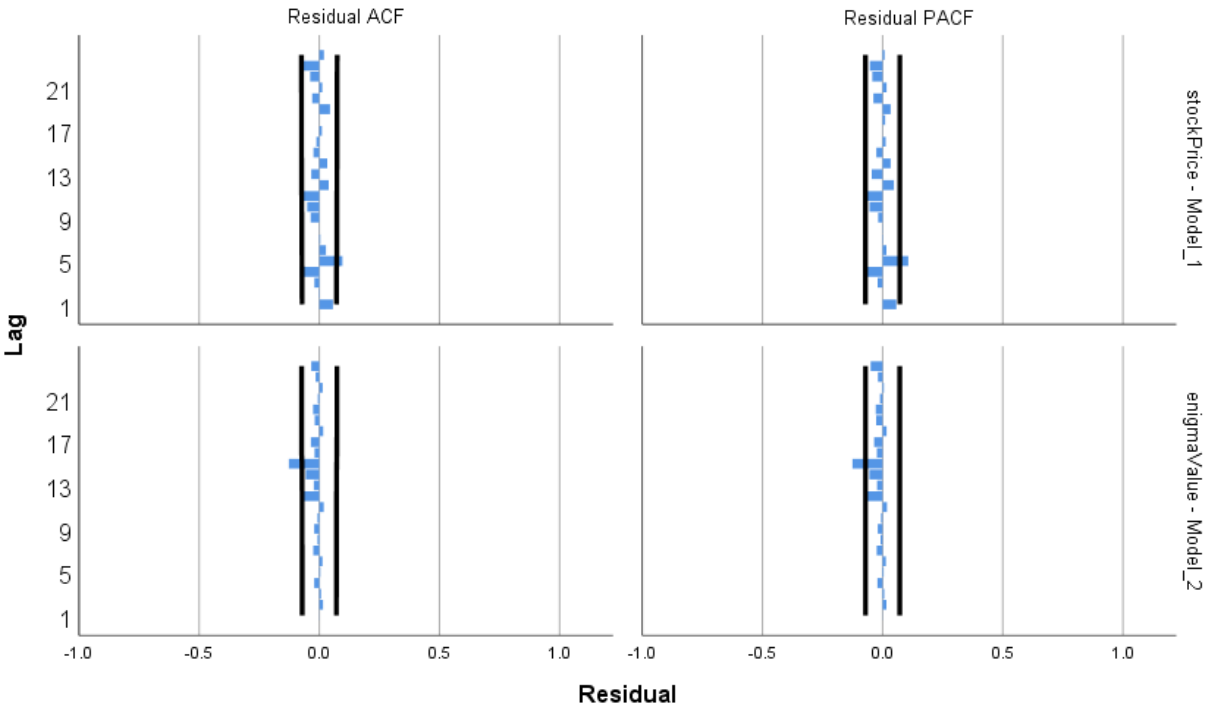
Residual PACF

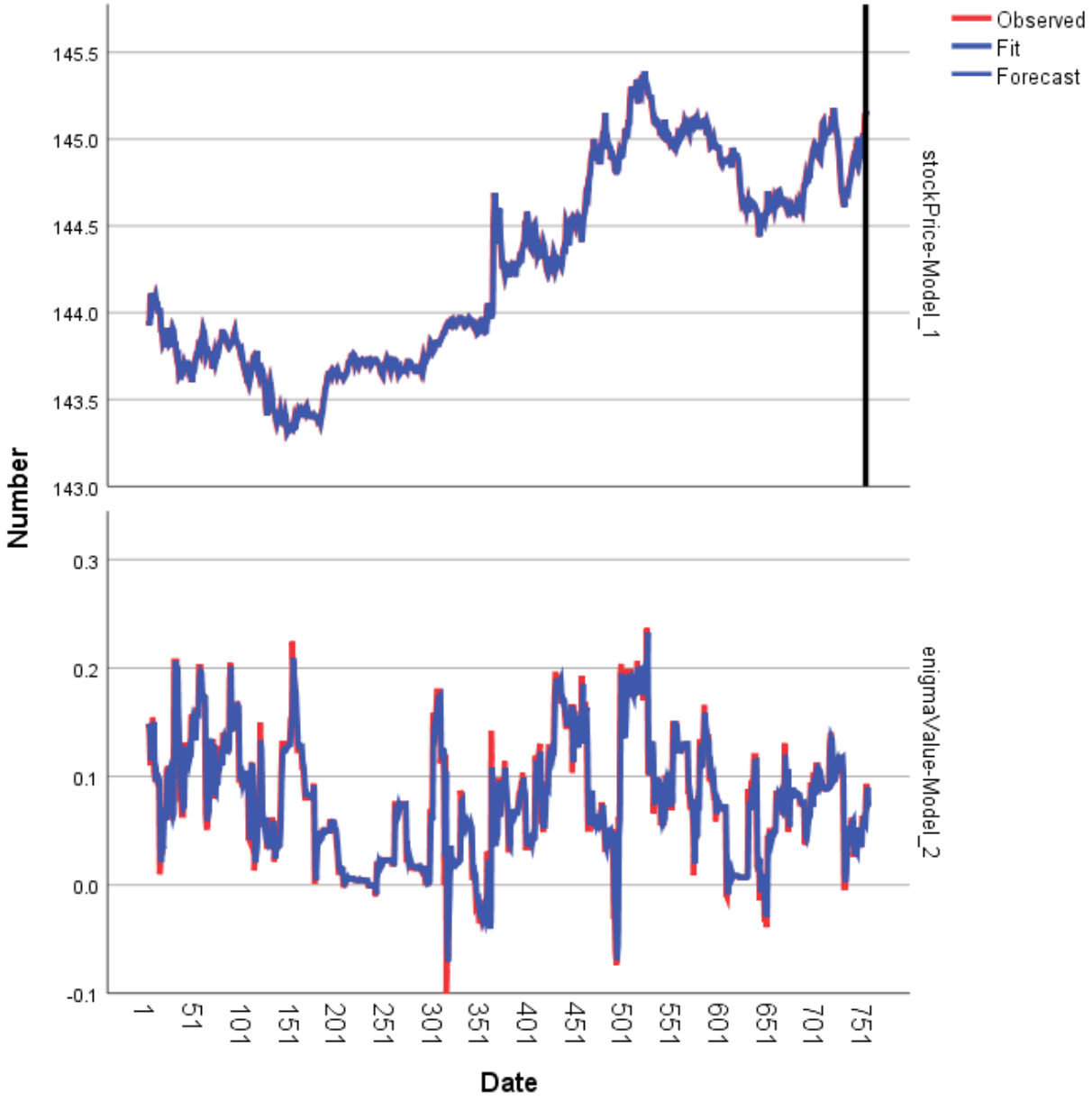
Model		19	20	21	22	23	24
stockPrice-Model_1	PACF	.034	-.039	.016	-.043	-.053	.008
	SE	.036	.036	.036	.036	.036	.036
enigmaValue-Model_2	PACF	-.027	-.028	-.011	.005	-.019	-.050
	SE	.036	.036	.036	.036	.036	.036

Forecast

Model		752	753	754	755	756
stockPrice-Model_1	Forecast	145.15	145.15	145.15	145.15	145.15
	UCL	145.24	145.27	145.30	145.32	145.34
	LCL	145.06	145.03	145.00	144.98	144.96
enigmaValue-Model_2	Forecast	.	.	.	.	.
	UCL	.	.	.	.	.
	LCL	.	.	.	.	.

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.


Residual ACF
Residual PACF
stockPrice - Model_1
enigmaValue - Model_2
Lag
21
17
13
9
5
1
-1.0
-0.5
0.0
0.5
1.0
Residual



PREDICT THRU END.

* Time Series Modeler.

TSMODEL

/MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

/MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

/MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=stockPrice enigmaValue
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:25:08
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

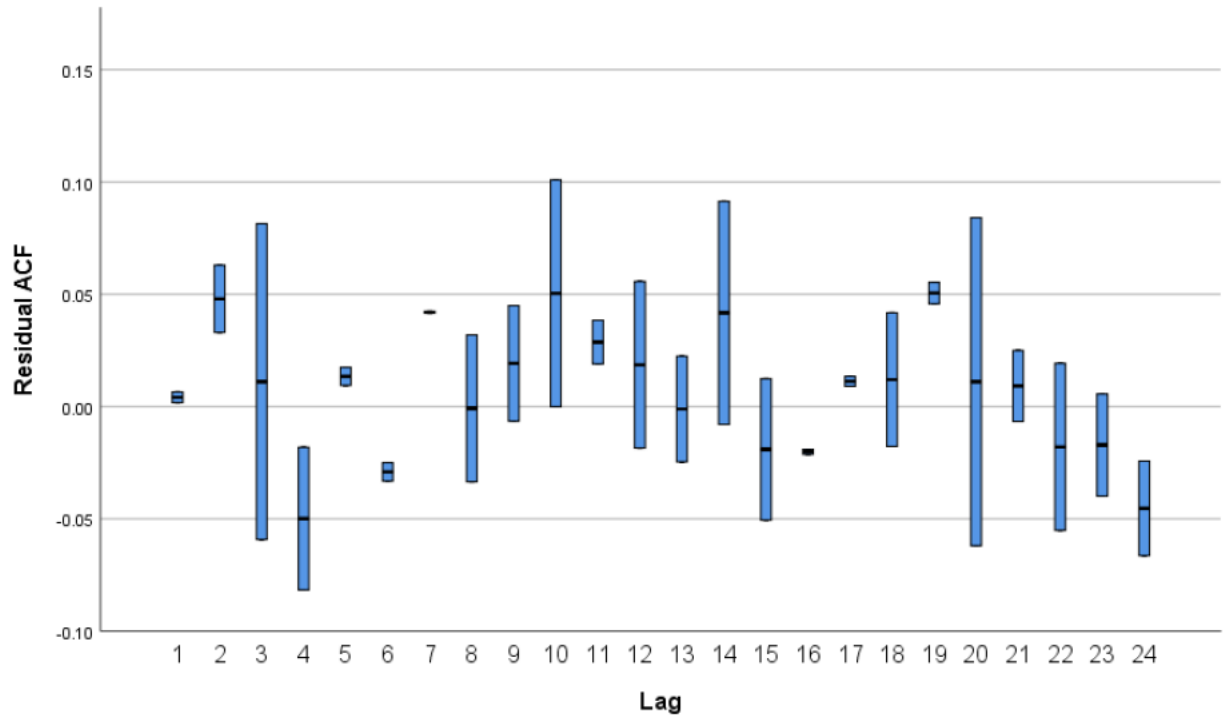
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=stockPrice enigmaValue PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.44
	Elapsed Time	00:00:00.72

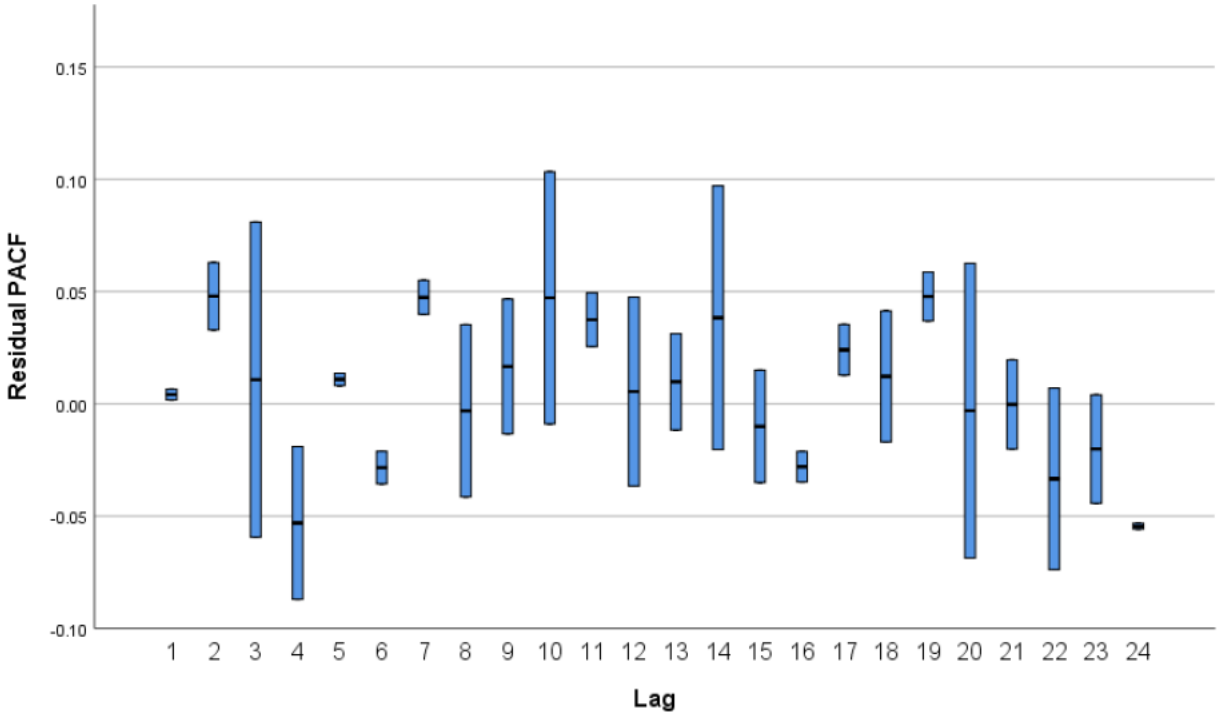
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	stockPrice	Model_1	ARIMA(1,1,0)
	enigmaValue	Model_2	ARIMA(1,0,1)

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.252	.330	.018	.486	.018	.018
R-squared	.734	.351	.486	.982	.486	.486
RMSE	.055	.011	.047	.063	.047	.047

[redacted] [redacted]

MAPE	79.083	111.803	.026	158.139	.026	.026
MaxAPE	4719.140	6673.017	.604	9437.675	.604	.604
MAE	.037	.012	.029	.046	.029	.029
MaxAE	.473	.260	.289	.657	.289	.289
Normalized BIC	-5.800	.423	-6.099	-5.501	-6.099	-6.099

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	.018	.252	.486	.486	.486
R-squared	.486	.734	.982	.982	.982
RMSE	.047	.055	.063	.063	.063
MAPE	.026	79.083	158.139	158.139	158.139
MaxAPE	.604	4719.140	9437.675	9437.675	9437.675
MAE	.029	.037	.046	.046	.046
MaxAE	.289	.473	.657	.657	.657
Normalized BIC	-6.099	-5.800	-5.501	-5.501	-5.501

Residual ACF Summary

					Percentile		
Lag	Mean	SE	Minimum	Maximum	5	10	25
Lag 1	.004	.003	.002	.007	.002	.002	.002
Lag 2	.048	.021	.033	.063	.033	.033	.033
Lag 3	.011	.099	-.059	.081	-.059	-.059	-.059
Lag 4	-.050	.045	-.082	-.018	-.082	-.082	-.082
Lag 5	.013	.006	.009	.017	.009	.009	.009
Lag 6	-.029	.006	-.033	-.025	-.033	-.033	-.033

[redacted] [redacted]

Lag 7	.042	.001	.042	.042	.042	.042	.042
Lag 8	-.001	.046	-.033	.032	-.033	-.033	-.033
Lag 9	.019	.036	-.006	.045	-.006	-.006	-.006
Lag 10	.050	.071	-4.793E-5	.101	-4.793E-5	-4.793E-5	-4.793E-5
Lag 11	.029	.014	.019	.038	.019	.019	.019
Lag 12	.019	.052	-.018	.056	-.018	-.018	-.018
Lag 13	-.001	.033	-.025	.022	-.025	-.025	-.025
Lag 14	.042	.070	-.008	.091	-.008	-.008	-.008
Lag 15	-.019	.044	-.051	.012	-.051	-.051	-.051
Lag 16	-.020	.001	-.021	-.019	-.021	-.021	-.021
Lag 17	.011	.003	.009	.013	.009	.009	.009
Lag 18	.012	.042	-.018	.042	-.018	-.018	-.018
Lag 19	.051	.007	.046	.055	.046	.046	.046
Lag 20	.011	.103	-.062	.084	-.062	-.062	-.062
Lag 21	.009	.022	-.007	.025	-.007	-.007	-.007
Lag 22	-.018	.053	-.055	.019	-.055	-.055	-.055
Lag 23	-.017	.032	-.040	.006	-.040	-.040	-.040
Lag 24	-.045	.030	-.066	-.024	-.066	-.066	-.066

Residual ACF Summary

	Percentile			
Lag	50	75	90	95
Lag 1	.004	.007	.007	.007
Lag 2	.048	.063	.063	.063
Lag 3	.011	.081	.081	.081
Lag 4	-.050	-.018	-.018	-.018
Lag 5	.013	.017	.017	.017
Lag 6	-.029	-.025	-.025	-.025

[redacted] [redacted]

Lag 7	.042	.042	.042	.042
Lag 8	-.001	.032	.032	.032
Lag 9	.019	.045	.045	.045
Lag 10	.050	.101	.101	.101
Lag 11	.029	.038	.038	.038
Lag 12	.019	.056	.056	.056
Lag 13	-.001	.022	.022	.022
Lag 14	.042	.091	.091	.091
Lag 15	-.019	.012	.012	.012
Lag 16	-.020	-.019	-.019	-.019
Lag 17	.011	.013	.013	.013
Lag 18	.012	.042	.042	.042
Lag 19	.051	.055	.055	.055
Lag 20	.011	.084	.084	.084
Lag 21	.009	.025	.025	.025
Lag 22	-.018	.019	.019	.019
Lag 23	-.017	.006	.006	.006
Lag 24	-.045	-.024	-.024	-.024

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	.004	.003	.002	.007	.002	.002	.002	.004
Lag 2	.048	.021	.033	.063	.033	.033	.033	.048
Lag 3	.011	.099	-.059	.081	-.059	-.059	-.059	.011
Lag 4	-.053	.048	-.087	-.019	-.087	-.087	-.087	-.053

[redacted] [redacted]

Lag 5	.011	.004	.008	.014	.008	.008	.008	.011
Lag 6	-.028	.010	-.036	-.021	-.036	-.036	-.036	-.028
Lag 7	.047	.011	.040	.055	.040	.040	.040	.047
Lag 8	-.003	.054	-.041	.035	-.041	-.041	-.041	-.003
Lag 9	.017	.042	-.013	.047	-.013	-.013	-.013	.017
Lag 10	.047	.079	-.009	.103	-.009	-.009	-.009	.047
Lag 11	.037	.017	.025	.049	.025	.025	.025	.037
Lag 12	.005	.059	-.037	.047	-.037	-.037	-.037	.005
Lag 13	.010	.030	-.012	.031	-.012	-.012	-.012	.010
Lag 14	.038	.083	-.020	.097	-.020	-.020	-.020	.038
Lag 15	-.010	.035	-.035	.015	-.035	-.035	-.035	-.010
Lag 16	-.028	.010	-.035	-.021	-.035	-.035	-.035	-.028
Lag 17	.024	.016	.013	.035	.013	.013	.013	.024
Lag 18	.012	.041	-.017	.041	-.017	-.017	-.017	.012
Lag 19	.048	.015	.037	.059	.037	.037	.037	.048
Lag 20	-.003	.093	-.069	.063	-.069	-.069	-.069	-.003
Lag 21	.000	.028	-.020	.020	-.020	-.020	-.020	.000
Lag 22	-.033	.057	-.074	.007	-.074	-.074	-.074	-.033
Lag 23	-.020	.034	-.044	.004	-.044	-.044	-.044	-.020
Lag 24	-.055	.002	-.056	-.053	-.056	-.056	-.056	-.055

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	.007	.007	.007
Lag 2	.063	.063	.063
Lag 3	.081	.081	.081
Lag 4	-.019	-.019	-.019

[redacted] [redacted]

Lag 5	.014	.014	.014
Lag 6	-.021	-.021	-.021
Lag 7	.055	.055	.055
Lag 8	.035	.035	.035
Lag 9	.047	.047	.047
Lag 10	.103	.103	.103
Lag 11	.049	.049	.049
Lag 12	.047	.047	.047
Lag 13	.031	.031	.031
Lag 14	.097	.097	.097
Lag 15	.015	.015	.015
Lag 16	-.021	-.021	-.021
Lag 17	.035	.035	.035
Lag 18	.041	.041	.041
Lag 19	.059	.059	.059
Lag 20	.063	.063	.063
Lag 21	.020	.020	.020
Lag 22	.007	.007	.007
Lag 23	.004	.004	.004
Lag 24	-.053	-.053	-.053

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
stockPrice-Model_1	0	.018	22.972	17	.150

[redacted] [redacted]

enigmaValue-Model_2	0	.486	24.981	16	.070

Model Statistics

Model	Number of Outliers
stockPrice-Model_1	0
enigmaValue-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
stockPrice-Model_1	ACF	.007	.063	.081	-.082	.017	-.025
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.002	.033	-.059	-.018	.009	-.033
	SE	.036	.036	.036	.037	.037	.037

Residual ACF

Model		7	8	9	10	11	12
stockPrice-Model_1	ACF	.042	-.033	.045	.000	.038	-.018
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	ACF	.042	.032	-.006	.101	.019	.056
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		13	14	15	16	17	18
stockPrice-Model_1	ACF	.022	-.008	-.051	-.019	.013	.042
	SE	.037	.037	.037	.038	.038	.038
enigmaValue-Model_2	ACF	-.025	.091	.012	-.021	.009	-.018

[redacted] [redacted]

	SE	.037	.037	.038	.038	.038	.038

Residual ACF

Model		19	20	21	22	23	24
stockPrice-Model_1	ACF	.055	.084	.025	-.055	-.040	-.066
	SE	.038	.038	.038	.038	.038	.038
enigmaValue-Model_2	ACF	.046	-.062	-.007	.019	.006	-.024
	SE	.038	.038	.038	.038	.038	.038

Residual PACF

Model		1	2	3	4	5	6
stockPrice-Model_1	PACF	.007	.063	.081	-.087	.008	-.021
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	.002	.033	-.059	-.019	.014	-.036
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		7	8	9	10	11	12
stockPrice-Model_1	PACF	.055	-.041	.047	-.009	.049	-.037
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	.040	.035	-.013	.103	.025	.047
	SE	.036	.036	.036	.036	.036	.036

Residual PACF

Model		13	14	15	16	17	18
stockPrice-Model_1	PACF	.031	-.020	-.035	-.035	.035	.041
	SE	.037	.037	.037	.037	.037	.037

[redacted] [redacted]

enigmaValue-Model_2	PACF	-.012	.097	.015	-.021	.013	-.017
	SE	.036	.036	.036	.036	.036	.036

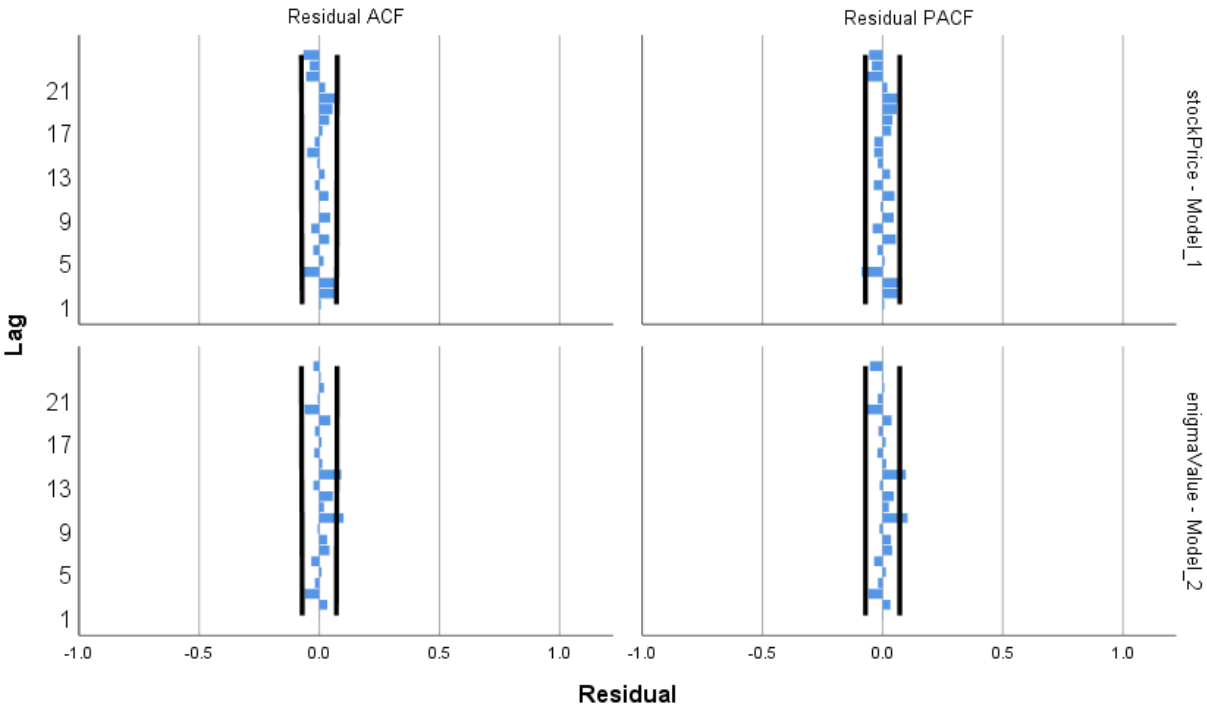
Residual PACF

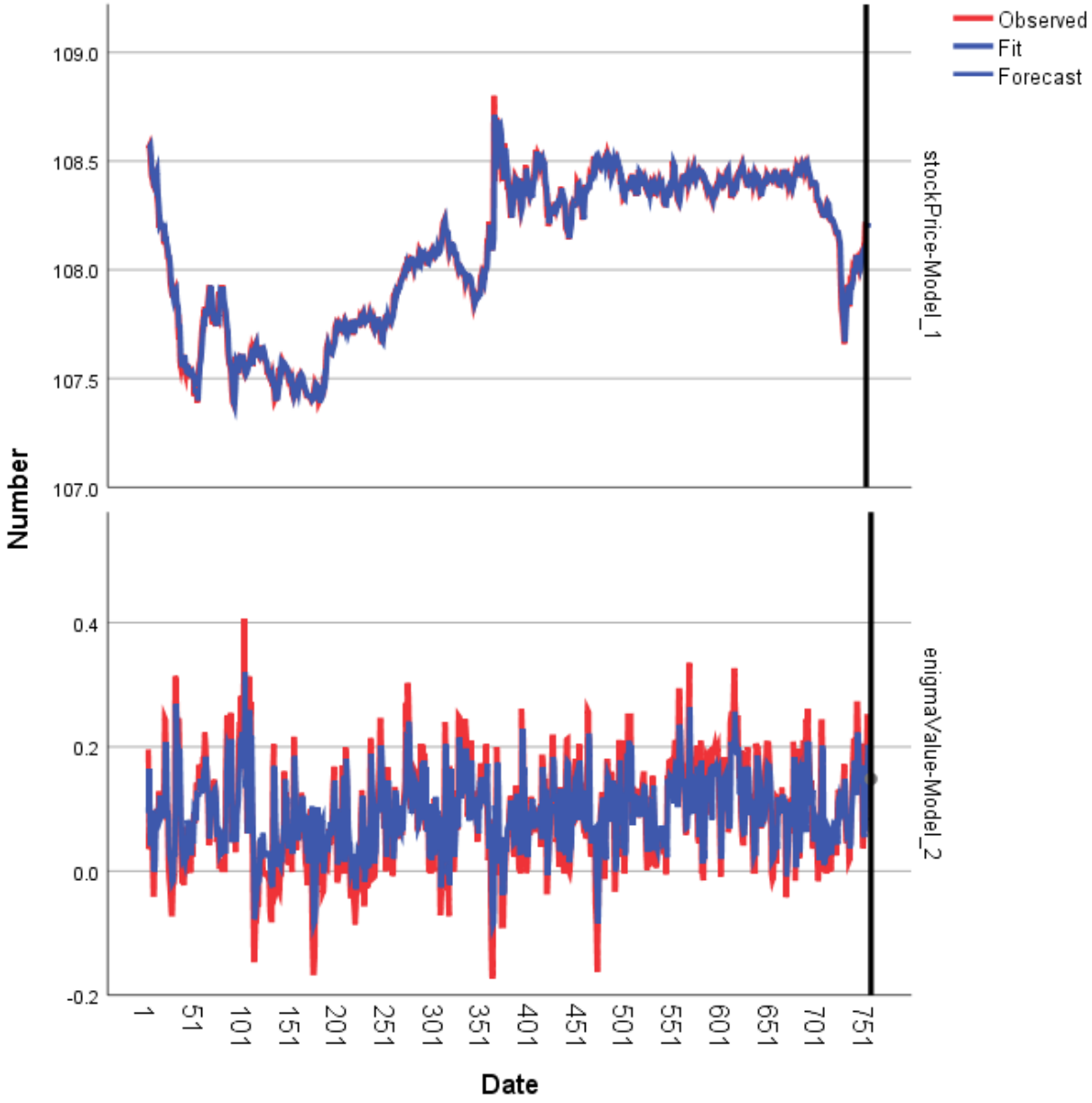
Model		19	20	21	22	23	24
stockPrice-Model_1	PACF	.059	.063	.020	-.074	-.044	-.056
	SE	.037	.037	.037	.037	.037	.037
enigmaValue-Model_2	PACF	.037	-.069	-.020	.007	.004	-.053
	SE	.036	.036	.036	.036	.036	.036

Forecast

Model		751	752	753	754	755	756
stockPrice-Model_1	Forecast	108.20	108.20	108.20	108.20	108.20	108.20
	UCL	108.29	108.33	108.35	108.37	108.39	108.41
	LCL	108.11	108.08	108.06	108.04	108.02	108.00
enigmaValue-Model_2	Forecast	.	.	.	.	.	.15
	UCL	.	.	.	.	.	.27
	LCL	.	.	.	.	.	.02

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.


Residual ACF
Residual PACF
stockPrice - Model_1
enigmaValue - Model_2
Lag
21
17
13
9
5
1
-1.0
-0.5
0.0
0.5
1.0
Residual



PREDICT THRU END.

* Time Series Modeler.

TSMODEL

/MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF]

/MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE]

/MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF]

[redacted] [redacted]

```
/SERIESPLOT OBSERVED FORECAST FIT
/OUTPUTFILTER DISPLAY=ALLMODELS
/AUXILIARY  CILEVEL=95 MAXACFLAGS=24
/MISSING USERMISSING=INCLUDE
/MODEL DEPENDENT=enigmaValue stockPrice
   PREFIX='Model'
/EXPERTMODELER TYPE=[ARIMA EXSMOOTH]
/AUTOOUTLIER  DETECT=OFF.
```

Time Series Modeler

Notes

Output Created		17-APR-2019 08:27:30
Comments		
Input	Active Dataset	DataSet0
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	756
	Date	<none>
Missing Value Handling	Definition of Missing	User-defined missing values are treated as valid data.

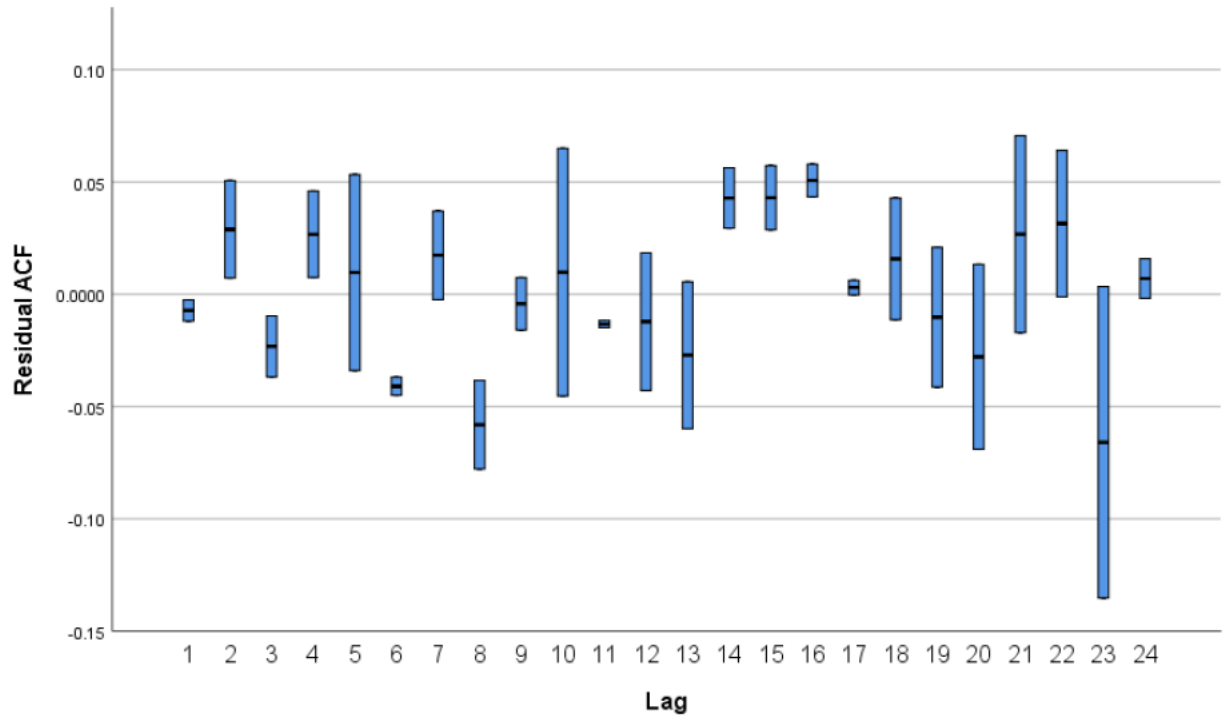
	Cases Used	Only cases with valid data for the dependent variable are used in computing any statistics.
Syntax		TSMODEL /MODELSUMMARY PRINT=[MODELFIT RESIDACF RESIDPACF] PLOT=[RESIDACF RESIDPACF] /MODELSTATISTICS DISPLAY=YES MODELFIT=[SRSQUARE] /MODELDETAILS PRINT=[RESIDACF RESIDPACF FORECASTS] PLOT=[RESIDACF RESIDPACF] /SERIESPLOT OBSERVED FORECAST FIT /OUTPUTFILTER DISPLAY=ALLMODELS /AUXILIARY CILEVEL=95 MAXACFLAGS=24 /MISSING USERMISSING=INCLUDE /MODEL DEPENDENT=enigmaValue stockPrice PREFIX='Model' /EXPERTMODELER TYPE=[ARIMA EXSMOOTH] /AUTOOUTLIER DETECT=OFF.
Resources	Processor Time	00:00:00.41
	Elapsed Time	00:00:00.41

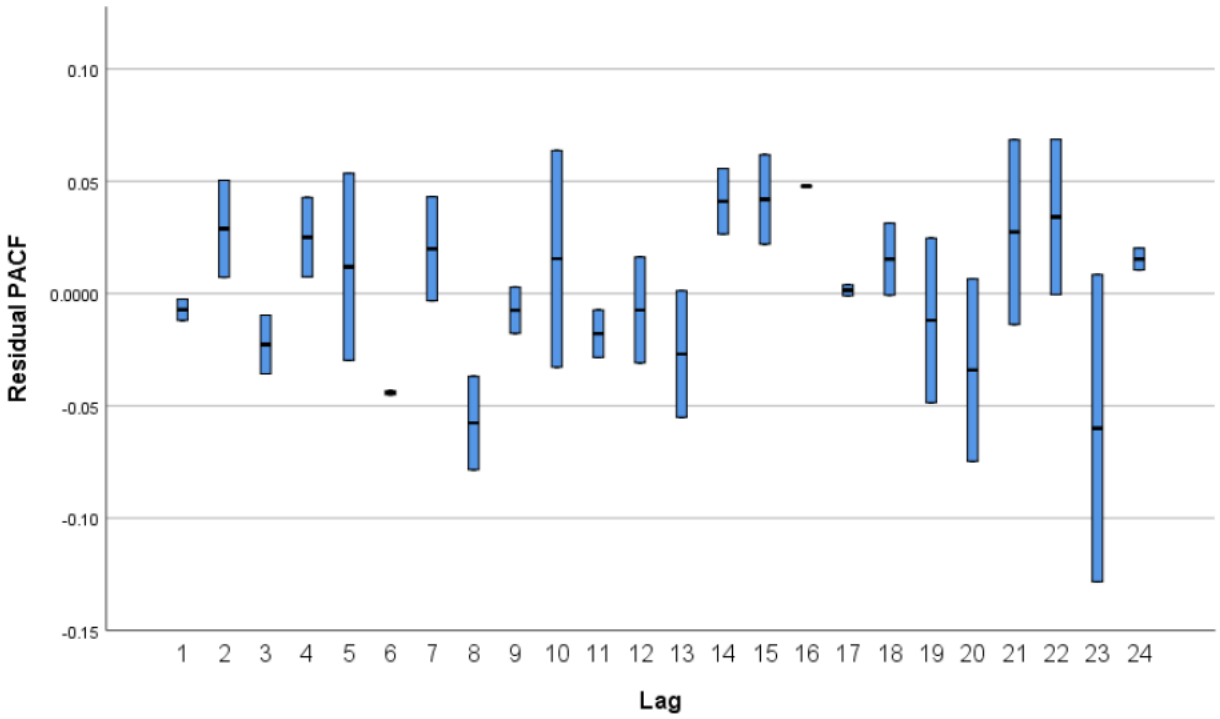
Use	From	First observation
	To	Last observation
Predict	From	First observation
	To	Last observation

Model Description

			Model Type
Model ID	enigmaValue	Model_1	ARIMA(2,0,0)
	stockPrice	Model_2	ARIMA(0,1,3)

Model Summary Chart





Model Summary

Model Fit

Fit Statistic	Mean	SE	Minimum	Maximum	Percentile 5	Percentile 10
Stationary R-squared	.358	.489	.012	.704	.012	.012
R-squared	.851	.209	.704	.999	.704	.704
RMSE	.115	.102	.043	.188	.043	.043

[redacted] [redacted]

MAPE	88.950	125.725	.049	177.852	.049	.049
MaxAPE	15254.773	21572.892	.435	30509.111	.435	.435
MAE	.079	.074	.027	.132	.027	.027
MaxAE	.732	.617	.296	1.169	.296	.296
Normalized BIC	-4.797	2.077	-6.266	-3.328	-6.266	-6.266

Model Fit

	Percentile				
Fit Statistic	25	50	75	90	95
Stationary R-squared	.012	.358	.704	.704	.704
R-squared	.704	.851	.999	.999	.999
RMSE	.043	.115	.188	.188	.188
MAPE	.049	88.950	177.852	177.852	177.852
MaxAPE	.435	15254.773	30509.111	30509.111	30509.111
MAE	.027	.079	.132	.132	.132
MaxAE	.296	.732	1.169	1.169	1.169
Normalized BIC	-6.266	-4.797	-3.328	-3.328	-3.328

Residual ACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	-.007	.007	-.012	-.003	-.012	-.012	-.012	-.007
Lag 2	.029	.031	.007	.051	.007	.007	.007	.029
Lag 3	-.023	.019	-.037	-.010	-.037	-.037	-.037	-.023
Lag 4	.027	.027	.007	.046	.007	.007	.007	.027
Lag 5	.010	.062	-.034	.053	-.034	-.034	-.034	.010
Lag 6	-.041	.006	-.045	-.037	-.045	-.045	-.045	-.041

[redacted] [redacted]

Lag 7	.017	.028	-.002	.037	-.002	-.002	-.002	.017
Lag 8	-.058	.028	-.078	-.038	-.078	-.078	-.078	-.058
Lag 9	-.004	.017	-.016	.007	-.016	-.016	-.016	-.004
Lag 10	.010	.078	-.045	.065	-.045	-.045	-.045	.010
Lag 11	-.013	.002	-.015	-.012	-.015	-.015	-.015	-.013
Lag 12	-.012	.043	-.043	.018	-.043	-.043	-.043	-.012
Lag 13	-.027	.046	-.060	.006	-.060	-.060	-.060	-.027
Lag 14	.043	.019	.029	.056	.029	.029	.029	.043
Lag 15	.043	.020	.029	.057	.029	.029	.029	.043
Lag 16	.051	.010	.043	.058	.043	.043	.043	.051
Lag 17	.003	.005	.000	.006	.000	.000	.000	.003
Lag 18	.016	.038	-.011	.043	-.011	-.011	-.011	.016
Lag 19	-.010	.044	-.041	.021	-.041	-.041	-.041	-.010
Lag 20	-.028	.058	-.069	.013	-.069	-.069	-.069	-.028
Lag 21	.027	.062	-.017	.071	-.017	-.017	-.017	.027
Lag 22	.031	.046	-.001	.064	-.001	-.001	-.001	.031
Lag 23	-.066	.098	-.135	.003	-.135	-.135	-.135	-.066
Lag 24	.007	.012	-.002	.016	-.002	-.002	-.002	.007

Residual ACF Summary

	Percentile		
Lag	75	90	95
Lag 1	-.003	-.003	-.003
Lag 2	.051	.051	.051
Lag 3	-.010	-.010	-.010
Lag 4	.046	.046	.046
Lag 5	.053	.053	.053
Lag 6	-.037	-.037	-.037

[redacted] [redacted]

Lag 7	.037	.037	.037
Lag 8	-.038	-.038	-.038
Lag 9	.007	.007	.007
Lag 10	.065	.065	.065
Lag 11	-.012	-.012	-.012
Lag 12	.018	.018	.018
Lag 13	.006	.006	.006
Lag 14	.056	.056	.056
Lag 15	.057	.057	.057
Lag 16	.058	.058	.058
Lag 17	.006	.006	.006
Lag 18	.043	.043	.043
Lag 19	.021	.021	.021
Lag 20	.013	.013	.013
Lag 21	.071	.071	.071
Lag 22	.064	.064	.064
Lag 23	.003	.003	.003
Lag 24	.016	.016	.016

Residual PACF Summary

					Percentile			
Lag	Mean	SE	Minimum	Maximum	5	10	25	50
Lag 1	-.007	.007	-.012	-.003	-.012	-.012	-.012	-.007
Lag 2	.029	.031	.007	.050	.007	.007	.007	.029
Lag 3	-.023	.018	-.036	-.010	-.036	-.036	-.036	-.023
Lag 4	.025	.025	.007	.043	.007	.007	.007	.025

[redacted] [redacted]

Lag 5	.012	.059	-.030	.054	-.030	-.030	-.030	.012
Lag 6	-.044	.001	-.045	-.043	-.045	-.045	-.045	-.044
Lag 7	.020	.033	-.003	.043	-.003	-.003	-.003	.020
Lag 8	-.058	.029	-.078	-.037	-.078	-.078	-.078	-.058
Lag 9	-.007	.015	-.018	.003	-.018	-.018	-.018	-.007
Lag 10	.015	.068	-.033	.064	-.033	-.033	-.033	.015
Lag 11	-.018	.015	-.028	-.007	-.028	-.028	-.028	-.018
Lag 12	-.007	.033	-.031	.016	-.031	-.031	-.031	-.007
Lag 13	-.027	.040	-.055	.001	-.055	-.055	-.055	-.027
Lag 14	.041	.021	.026	.056	.026	.026	.026	.041
Lag 15	.042	.028	.022	.062	.022	.022	.022	.042
Lag 16	.048	.000	.048	.048	.048	.048	.048	.048
Lag 17	.001	.003	-.001	.004	-.001	-.001	-.001	.001
Lag 18	.015	.023	-.001	.031	-.001	-.001	-.001	.015
Lag 19	-.012	.052	-.049	.025	-.049	-.049	-.049	-.012
Lag 20	-.034	.057	-.075	.007	-.075	-.075	-.075	-.034
Lag 21	.027	.058	-.014	.068	-.014	-.014	-.014	.027
Lag 22	.034	.049	.000	.069	.000	.000	.000	.034
Lag 23	-.060	.097	-.128	.008	-.128	-.128	-.128	-.060
Lag 24	.015	.007	.010	.020	.010	.010	.010	.015

Residual PACF Summary

	Percentile		
Lag	75	90	95
Lag 1	-.003	-.003	-.003
Lag 2	.050	.050	.050
Lag 3	-.010	-.010	-.010
Lag 4	.043	.043	.043

[redacted] [redacted]

Lag 5	.054	.054	.054
Lag 6	-.043	-.043	-.043
Lag 7	.043	.043	.043
Lag 8	-.037	-.037	-.037
Lag 9	.003	.003	.003
Lag 10	.064	.064	.064
Lag 11	-.007	-.007	-.007
Lag 12	.016	.016	.016
Lag 13	.001	.001	.001
Lag 14	.056	.056	.056
Lag 15	.062	.062	.062
Lag 16	.048	.048	.048
Lag 17	.004	.004	.004
Lag 18	.031	.031	.031
Lag 19	.025	.025	.025
Lag 20	.007	.007	.007
Lag 21	.068	.068	.068
Lag 22	.069	.069	.069
Lag 23	.008	.008	.008
Lag 24	.020	.020	.020

Model Statistics

Model	Number of Predictors	Model Fit statistics: Stationary R-squared	Ljung-Box Q(18): Statistics	DF	Sig.
enigmaValue-Model_1	0	.704	24.470	16	.080

[redacted] [redacted]

stockPrice-Model_2	0	.012	14.352	16	.572

Model Statistics

Model	Number of Outliers
enigmaValue-Model_1	0
stockPrice-Model_2	0

Residual ACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	ACF	-.012	.051	-.037	.046	-.034	-.037
	SE	.036	.036	.037	.037	.037	.037
stockPrice-Model_2	ACF	-.003	.007	-.010	.007	.053	-.045
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		7	8	9	10	11	12
enigmaValue-Model_1	ACF	.037	-.078	.007	-.045	-.015	-.043
	SE	.037	.037	.037	.037	.037	.037
stockPrice-Model_2	ACF	-.002	-.038	-.016	.065	-.012	.018
	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	ACF	.006	.056	.057	.058	.000	.043
	SE	.037	.037	.037	.037	.038	.038
stockPrice-Model_2	ACF	-.060	.029	.029	.043	.006	-.011

[redacted] [redacted]

	SE	.037	.037	.037	.037	.037	.037

Residual ACF

Model		19	20	21	22	23	24
enigmaValue-Model_1	ACF	.021	.013	-.017	-.001	.003	-.002
	SE	.038	.038	.038	.038	.038	.038
stockPrice-Model_2	ACF	-.041	-.069	.071	.064	-.135	.016
	SE	.037	.037	.037	.038	.038	.038

Residual PACF

Model		1	2	3	4	5	6
enigmaValue-Model_1	PACF	-.012	.050	-.036	.043	-.030	-.043
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.003	.007	-.010	.007	.054	-.045
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		7	8	9	10	11	12
enigmaValue-Model_1	PACF	.043	-.078	.003	-.033	-.028	-.031
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.003	-.037	-.018	.064	-.007	.016
	SE	.037	.037	.037	.037	.037	.037

Residual PACF

Model		13	14	15	16	17	18
enigmaValue-Model_1	PACF	.001	.056	.062	.048	-.001	.031
	SE	.036	.036	.036	.036	.036	.036

stockPrice-Model_2	PACF	-.055	.026	.022	.048	.004	-.001
	SE	.037	.037	.037	.037	.037	.037

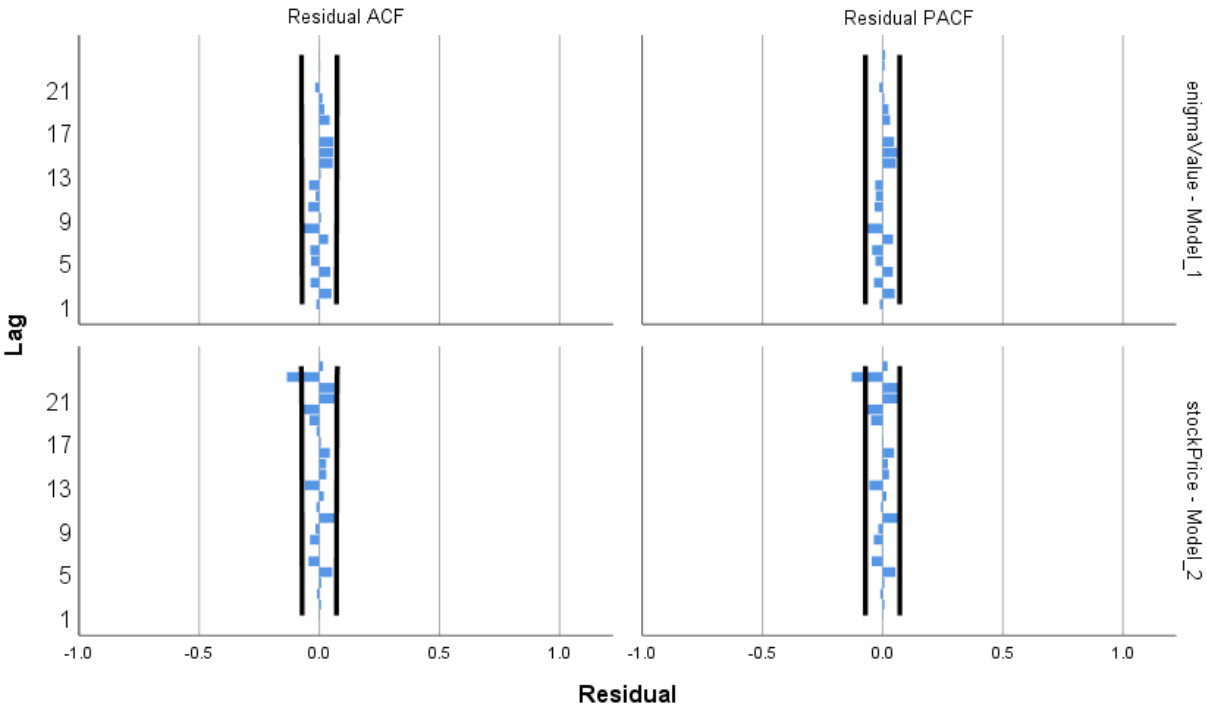
Residual PACF

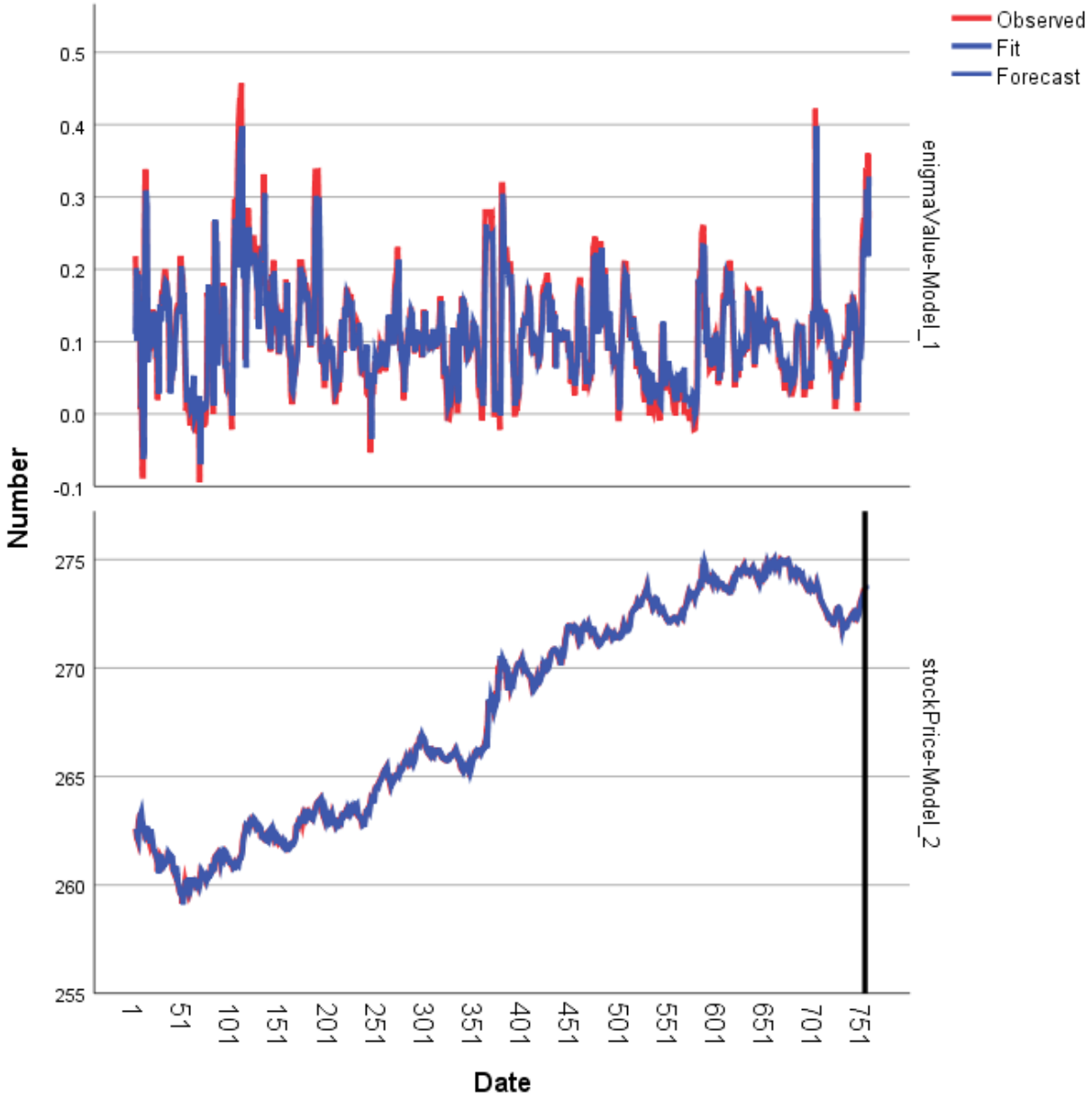
Model		19	20	21	22	23	24
enigmaValue-Model_1	PACF	.025	.007	-.014	.000	.008	.010
	SE	.036	.036	.036	.036	.036	.036
stockPrice-Model_2	PACF	-.049	-.075	.068	.069	-.128	.020
	SE	.037	.037	.037	.037	.037	.037

Forecast

Model		752	753	754	755	756
enigmaValue-Model_1	Forecast	.	.	.	.	.
	UCL	.	.	.	.	.
	LCL	.	.	.	.	.
stockPrice-Model_2	Forecast	273.69	273.71	273.72	273.72	273.72
	UCL	274.06	274.25	274.40	274.53	274.64
	LCL	273.32	273.16	273.05	272.92	272.80

For each model, forecasts start after the last non-missing in the range of the requested estimation period, and end at the last period for which non-missing values of all the predictors are available or at the end date of the requested forecast period, whichever is earlier.


Residual ACF
Residual PACF
enigmaValue - Model_1
stockPrice - Model_2
Lag
21
17
13
9
5
1
-1.0
-0.5
0.0
0.5
1.0
Residual


Observed
Fit
Forecast
Number
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
enigmaValue-Model_1
275
270
265
260
255
stockPrice-Model_2
1
51
101
151
201
251
301
351
401
451
501
551
601
651
701
751
Date